Virtuelle Mitglieder in virtuellen Teams

Modellierung, Umsetzung und Evaluation
eines analytischen Verfahrens
innerhalb einer CSCL-Umgebung

Vom Fachbereich III
(Informations- und Kommunikationswissenschaften)
der Universität Hildesheim zur Erlangung des Grades

eines Doktors der Philosophie
(Dr. phil.)

angenommene Dissertation von

Glenn Langemeier
geboren am 12.06.1967 in Alfeld/Leine

Hil 2

Gutachterin:	Prof. Dr. Christa Womser-Hacker
Gutachter:	Prof. Dr. Eberhard Schwarzer
Tag der mündlichen Prüfung:	15.02.2008

für meine Familie

Vorwort

Die vorliegende Arbeit entstand im Rahmen meiner Lehr- und Forschungstätigkeit am Institut für Physik und Technik (ehemals Institut für Physik und Technische Informatik) der Universität Hildesheim. Diese Tätigkeit begann im Wintersemester 1999 mit einem Lehrauftrag für die Veranstaltung *„Programmierung in Java (Einführung)"*, die ich über mehrere Jahre regelmäßig im Wintersemester in Form eines Programmierpraktikums als Erstsemesterveranstaltung im Bachelor-Studiengang *„Informationsmanagement und Informationstechnologie"* (IMIT Bsc) angeboten und durchgeführt habe. Aufgrund der dort zu beobachtenden Probleme, mit denen sich die Teilnehmer – in der Regel Anfänger in der Programmierung – bei der Bearbeitung von Programmieraufgaben konfrontiert sehen, und dem daraus resultierenden Bedarf an tutorieller Unterstützung entstand die Idee zu einem Unterstützungssystem, das die Teilnehmer bei Problemen der objektorientierten Programmierung in Java angemessen unterstützt, indem es angepassten Support bietet, so dass weder die Motivation der Lernenden durch zuwenig Hilfe noch der beabsichtigte Lerneffekt durch zuviel Unterstützung beeinträchtigt werden.

Im Rahmen des niedersächsischen ELAN-Projekts entstand das kollaborative Lernsystem *VitaminL* (Virtuelle Teams: Analyse und Modellierung in netzbasierten Lernumgebungen), das zur Durchführung von virtuellen Tutorien und Benutzertests verwendet wurde. VitaminL wurde in Kooperation mit dem Institut für Angewandte Sprachwissenschaft entwickelt. Einen wesentlichen Anteil an der Entwicklung hat mein Freund und Kollege Dr. Ralph Kölle, dem mein spezieller Dank gilt.

Meine beiden Gutachter und Betreuer, Frau Prof. Dr. Christa Womser-Hacker und Herr Prof. Dr. Eberhard Schwarzer, haben entscheidend zum Gelingen dieser Arbeit beigetragen. Sie haben vom Prozess der Themenfindung bis hin zur Fertigstellung dieser Arbeit stets in gleichem Maße wertvolle Anregungen und Hinweise gegeben wie auch konstruktive Kritik geübt. Ihnen möchte ich hiermit besonders danken.

Ein großes Dankeschön geht auch an Herrn Prof. Dr. Stephan Schlickau, der freundlicherweise den Vorsitz der Prüfungskommission übernommen hat. Danken möchte ich auch den Mitarbeiterinnen und Mitarbeitern des Dekanats für die organisatorischen Hilfestellungen.

Weiterer Dank gebührt dem Leiter des Instituts für Physik und Technik, Herrn

Prof. Dr.-Ing. Jürgen-Rüdiger Böhmer, der mir nicht nur neue, interessante Aufgaben und Verantwortungsbereiche übertrug, sondern mir auch gleichzeitig den notwendigen Freiraum gab, um diese Arbeit erfolgreich fertigzustellen.

Danken möchte ich auch meinen Kolleginnen und Kollegen, die mit anregenden Diskussionen, nützlichen Hinweisen und einem angenehmen Arbeitsklima ihren Beitrag zu dieser Arbeit geleistet haben. Besonders zu erwähnen sind an dieser Stelle die Teilnehmer des Promotionsnetzwerks sowie die Mitglieder des Instituts für Physik und Technik.

Zu Dank verpflichtet fühle ich mich auch den Studierenden, die an Benutzertests teilgenommen haben oder mit Projekt- und Abschlussarbeiten ihren individuellen Beitrag zum VitaminL-Projekt geleistet haben.

Ein großes Dankeschön geht an meine 'Lektorin' Frau Julia Schlüter, die mit jedem gefundenen Rechtschreibfehler zur Verbesserung dieser Arbeit beigetragen hat.

Mein herzlichster Dank gebührt meiner Familie, meinen Freunden und meinen Volleyballern, die mich während meiner Promotionszeit entweder ertragen oder entbehren mussten. Ich danke meinen Eltern, die mich auf meinem ganzen Weg unterstützt haben. Mein Dank geht auch an meine Schwiegereltern für ihre stete Unterstützung. Ganz besonders danke ich meiner Frau Isabell, die mir während der ganzen Zeit stets den Rücken freigehalten und die somit diese Arbeit überhaupt erst ermöglicht hat. Ihr und unseren beiden Kindern, Helena und Dominik, ist diese Arbeit gewidmet.

Hildesheim, im Februar 2008 Glenn Langemeier

Zusammenfassung

Kooperatives Lernen zeichnet sich gegenüber dem individuellen Lernen durch eine Vielzahl von Vorteilen aus. Diese äußern sich vornehmlich in einer potentiell erhöhten Lernmotivation, welche sich wiederum in einem positiveren Lernerfolg niederschlägt. Darüber hinaus bietet diese Lernform Möglichkeiten zum Erwerb von Sozialkompetenz und weiteren wichtigen Schlüsselkompetenzen wie beispielsweise Kommunikations- und Kooperationskompetenz.

Diese Vorteile macht man sich auch in Vorlesungen zur Programmierung zunutze, indem die Vertiefung der theoretischen Lerninhalte mittels praktischer Übungen und Projektarbeiten in Gruppen erfolgt. Dabei erfahren die Gruppen bei Bedarf Hilfe durch einen Tutor. Unterstützt man dieses Szenario unter Verwendung moderner Informations- und Kommunikationstechnologien, insbesondere durch die Kombination von Gruppeneditoren, Chat-Komponenten und weiteren Werkzeugen der computer-vermittelten Kommunikation, so bedeutet dieses einen Übergang vom traditionellen Lernen hin zu einem E-Learning-Szenario. Aus dem Tutor wird ein Teletutor, aus den Gruppen werden virtuelle Teams, die nunmehr unter zeitlicher und räumlicher Entkopplung miteinander arbeiten und (programmieren) lernen. Führt man diese Entkopplung weiter fort, wird der (menschliche) Teletutor gegen einen virtuellen Tutor substituiert, der – in intelligenten tutoriellen Systemen als Spezialform des E-Learning – dem Lernenden, angepasst an dessen Wissensstand, Unterstützung anbietet. Für die Unterstützung virtueller Teams bedeutet dies eine Anpassung an Teamwissen anstelle von Individualwissen.

Das VitaminL-System als ein neuartiger Ansatz zur tutoriellen Unterstützung virtueller Teams geht bei der Modellierung von Teamwissen von einem Rollenmodell aus, das alle für die Zusammenarbeit wichtigen Teamfunktionen in Form von Rollen definiert. Es wird davon ausgegangen, dass alle Rollen – mit unter Umständen variierenden Gewichtungen – für die Zusammenarbeit relevant sind. Sollten sich dabei Problemsituationen ergeben, so greift üblicherweise der (virtuelle) Tutor ein und ergänzt dabei das Team um diejenige(n) Rolle(n), die innerhalb des Teams unterrepräsentiert ist (bzw. sind), um somit fehlende Funktionen zu kompensieren.

Die vorliegende Arbeit untersucht primär Möglichkeiten, um während der Zusammenarbeit die Ausprägungen aller Teammitglieder hinsichtlich der vorhandenen Rollen zu bestimmen, so dass die Tutorkomponente ihr Unterstützungsangebot an die Rollenzusammensetzung des jeweiligen virtuellen Teams anpassen kann.

Ergänzend dazu wurde eine Studie durchgeführt, die sich mit den für diese spezielle Form der Zusammenarbeit typischen Problemen befasst. Das Ziel bestand darin, ein Konzept für eine Problemidentifikationskomponente als Teil der Tutorkomponente zu entwickeln, mit deren Hilfe Problemsituationen während der Zusammenarbeit durch den virtuellen Tutor entdeckt werden können. Die Untersuchungen und Evaluationen sowohl zur Rollen- als auch zur Problemanalyse fanden im wesentlichen auf der Grundlage von Benutzertests und Befragungen statt.

Die Ergebnisse der Rollenanalyse bleiben zunächst hinter den ursprünglichen Erwartungen zurück, liefern aber bei genauerer Betrachtung interessante Ansätze zur Weiterentwicklung der Rollenanalyse. Ähnlich verhält es sich bei der Problemanalyse, die keine eindeutige Erkennung von Problemen liefert, dafür jedoch eine Menge an Problemindikatoren, die – zusammen mit weiteren Konzepten – die Grundlage für eine Evolution des ursprünglichen Analysekonzepts bilden. Insgesamt werden die erzielten Resultate mitsamt ihrer jeweiligen Weiterentwicklungskonzepte als Bestätigung des Konzepts einer rollenbasierten Tutorkomponente gewertet.

Inhalt

Tabellen

Abbildungen

Beispiele

Kapitel 1

Einleitung

Dem kooperativen Lernen werden gegenüber dem individuellen Lernen gewisse Vorteile bescheinigt: *„Viele Forschungsarbeiten haben demonstriert, dass Lernende, die kooperativ arbeiten, bessere Leistungen erreichen als jene, die dasselbe Material individuell lernen"* (Konrad & Traub, 2005, S.180). Als einer der wichtigsten Vorteile wird dabei die gemeinsame Konstruktion von Wissen angesehen: *„Die ausgehandelten gemeinsamen Interpretationen der Lernenden sind reichhaltiger, weil mehr Wissen berücksichtigt wird und verschiedene Perspektiven eingenommen werden"* (Konrad & Traub, 2005, S.180). Ähnlich sehen es Kriz & Nöbauer: *„Gruppen werden aber auch als wesentliche Elemente organisationalen Lernens betrachtet. ... Innerhalb und zwischen Gruppen wird nicht nur neues Wissen generiert, sondern auch vorhandenes Wissen vernetzt und zur Verfügung gestellt"* (Kriz & Nöbauer, 2002, S.20). Auch Pfister & Wessner gelangen zu dem Schluss, *„dass kooperative Lernformen dem individuellen Lernen oft überlegen sind"* (Pfister & Wessner, 2001a, S.7).

Aufgrund der rasanten Fortschritte der vergangenen Jahre in den Informations- und Kommunikationstechnologien konnten zahlreiche Systeme, sog. *Groupware*, entwickelt werden, die Gruppen bei ihrer Zusammenarbeit unter Zuhilfenahme miteinander vernetzter Computer unterstützen. Das diesen Systemen zugrundeliegende Forschungsgebiet wird als *Computer Supported Cooperative Work* (CSCW) bezeichnet. Auf interdisziplinärer Basis wird untersucht, wie innerhalb von Gruppen zusammengearbeitet wird und wie die Gruppenprozesse unter Zuhilfenahme der Informations- und Kommunikationstechnologien unterstützt werden können, um dabei die Effektivität und Effizienz der Gruppenarbeit zu steigern (vgl. Teufel et al., 1995, S.17; Koch & Gross, 2006 in Feltz et al., 2006). Das Lernen in der Gruppe steht im Mittelpunkt des *Computer Supported Cooperative Learning* (CSCL), das als Spezialgebiet des CSCW betrachtet werden kann (vgl. Wessner, 2001 in Schulmeister, 2001).

Das Spektrum der Computerunterstützung reicht von einfachen E-Learning-Plattformen, mit denen Kurse organisiert und Lernmaterialien verwaltet und angeboten werden können, bis hin zu *intelligenten tutoriellen Systemen* (ITS), die auch in der Lage sind, den Lernprozess einzelner Teilnehmer oder auch ganzer Gruppen

durch virtuelle Tutoren aktiv zu fördern (vgl. Kerres, 2001, S.71f; Schulmeister, 1997, S.177ff).

1.1 Problembeschreibung

Das Erlernen einer Programmiersprache wie beispielsweise der objektorientierten Sprache *Java* stellt Lernende vor besondere Herausforderungen – insbesondere dann, wenn sie keine Vorkenntnisse oder nur geringe Programmiererfahrungen besitzen. Neben den technischen Problemen, die sich aus der Notwendigkeit der Installation eines Programmiersystems und dessen Bedienung ergeben, müssen auch inhaltliche Probleme bewältigt werden: Sowohl das Erlernen von Syntax und Semantik einer formalen Sprache wie auch das Formulieren und Umsetzen von Problemlösungen mit solch einer Sprache stellen weitere Hürden dar. Trotz umfangreicher Literatur und ausführlichen Installationsanleitungen können die geschilderten Probleme oft nur durch das Hinzuziehen eines (realen) Tutors gelöst werden.

Auch im fortgeschrittenen Stadium einer Lehrveranstaltung zur Programmierung kann auf die weiterführende Hilfe eines Tutors in den allermeisten Fällen nicht verzichtet werden. An die Stelle von Problemen mit Syntax und Semantik der verwendeten Programmiersprache treten nun solche, die sich mit dem Design möglicher Problemlösungen befassen. Hier kann die Erfahrung des Tutors von großem Wert sein: *„Es dauert lange, bis Anfänger verstehen, worum es bei gutem objektorientiertem Entwurf geht. Offenkundig wissen erfahrene Entwickler etwas, was unerfahrene nicht wissen"* (Gamma et al., 1996, S.1).

Werden Programmierpraktika in Kleingruppen durchgeführt, können verschiedene positive Effekte beobachtet werden: Einerseits kann die Anzahl erforderlicher Tutoren relativ klein gehalten werden, da sich aufgrund der Gruppenbildung die Anzahl der Kontakte zwischen Lernenden und Tutor reduziert, andererseits wird vorhandenes Wissen zwischen den Gruppenmitgliedern aktiv ausgetauscht und verteilt. Es ergeben sich jedoch im Gegenzug organisatorische Probleme, da sich die Gruppenmitglieder sowohl für die Bearbeitung von Übungsaufgaben auf gemeinsame Termine und Treffpunkte einigen müssen als auch eine geeignete Verteilung der anfallenden Aufgaben und Arbeitspakete treffen müssen. Überdies ist die Verfügbarkeit realer Tutoren üblicherweise beschränkt auf ein paar wenige Präsenzstunden pro Woche. Diese Zeiten überschneiden sich in den seltensten Fällen mit den Zeiträumen, in denen die Lerngruppen tatsächlich tutoriellen Support benötigen. Ergänzende Unterstützung via Email und Online-Foren wird häufig durch die Komplexität der Fragen und Probleme erschwert. Als Folge davon muss die weitere Bearbeitung von Übungsaufgaben oft bis zum nächsten Tutorium vertagt werden.

Als Lösungsansatz für die geschilderten Probleme wird ein CSCL-System gefordert, das in der Lage ist, virtuelle Teams beim gemeinschaftlichen Erlernen einer Programmiersprache zu unterstützen. Da die funktionale Zusammensetzung der jeweiligen Gruppe für die effektive Aufgabenerfüllung relevant ist, sollte der virtuelle Tutor (als Komponente des CSCL-Systems) auf der Basis eines geeigneten

Rollenmodells die Gruppenzusammensetzung berücksichtigen und fehlende beziehungsweise unterrepräsentierte Rollen kompensieren.

1.2 Forschungslücken

Die Forschungen im Bereich CSCL haben bereits einige Systeme hervorgebracht, die produktiv zur Unterstützung von Lehrveranstaltungen eingesetzt werden können (vgl. SCHULMEISTER, 2001, S.165-194). So kann insbesondere die computer-gestützte Gruppenarbeit in virtuellen Teams mittels geeigneter Groupware beziehungsweise CSCW-/CSCL-Systeme einen Lösungsansatz für die oben genannten organisatorischen Probleme bieten.

Die Unterstützung des Lernprozesses durch CSCL-Systeme steckt jedoch immer noch in den Anfängen: *„Existierende Systeme berücksichtigen den Kontext des kooperativen Lernens nicht in ausreichendem Maße. Da das kooperative Lernen dem System gegenüber nicht durch Gruppenbeschreibung, Aufgabe, Methode und die Einordnung in den Lernprozess näher bestimmt ist, kann das System lediglich als passives, ermöglichendes Medium dieser Kooperation fungieren"* (WESSNER, 2005, S. 4). Diese fehlende Unterstützung des Lernprozesses durch ITS hat vor einiger Zeit bereits SCHULMEISTER kritisiert: *„Das Gebiet der intelligenten tutoriellen Systeme ist offenbar in der Phase der Programmatik stecken geblieben: Selbst wenn man nur die Aufsätze einer einzigen Zeitschrift zu dem Thema ITS betrachtet, so wiederholen sich ständig dieselben Aussagen, Systematisierungen und Rückgriffe auf die AI-Literatur. Ich habe selten so viel Redundanz auf einem Haufen gesehen, und selten eine so kleine Gemeinschaft von Forschern, die ständig dasselbe auf demselben Entwicklungsstand veröffentlichen."* (SCHULMEISTER, 1997, S.203). Es besteht also offensichtlich Bedarf, im Gebiet CSCL neue Ansätze der Unterstützung von Lerngruppen durch ein ITS zu untersuchen.

Die folgende Definiton von ITS offenbart gleich mehrere Schwachpunkte dieser Systeme: *„Intelligente Tutorielle Systeme (ITS) zeichnen sich dadurch aus, dass sie auf der Grundlage der Analyse des Lernerwissens adaptiv Aufgaben generieren bzw. Instruktionsschritte auswählen und folglich besonders flexibel auf die Eingaben des Lernenden reagieren"* (SCHAUMBURG, 2003, S.47). Zunächst einmal muss die Domäne hinreichend genau erfasst und modelliert werden. Ein ITS ist also stets auf ein klar umrissenes Problemgebiet beschränkt. Des weiteren muss ein solches System in der Lage sein, aus den Eingaben des Lernenden auf dessen Wissensstand zu schließen. Und letztlich wird verlangt, dass sich das System adaptiv verhält, sich also an die Bedürfnisse des Lernenden flexibel anpasst.

Trotz aller Kritik haben die Forschungen der vergangenen Jahre einige interessante Ansätze hervorgebracht, den Lernprozess mittels virtueller Tutoren zu fördern. DOMESHEK ET AL. verwenden einen *sokratischen Tutor* (vgl. STEVENS & COLLINS, 1977), der mit einem Lernenden in Dialog tritt, um mittels verschiedener kommunikativer Strategien den Lernenden selber eine Problemlösung erarbeiten zu lassen (vgl. DOMESHEK ET AL., 2004). *STEVE*, ein pädagogischer Agent, ist in

der Lage, mit einem Lernenden oder aber auch einer Gruppe von Lernenden Probleme aus vornehmlich technischen Bereichen (wie bspw. der korrekten Bedienung von Gasturbinenmotoren in U-Booten) zu lösen. Die komplette Lösung wird in Teilschritte und Beziehungen zwischen diesen Teilschritten (in Form kausaler Zusammenhänge) modelliert. Die Interaktion zwischen dem Lernendem und seinem Tutor *STEVE* findet in einer virtuellen Welt als Abbild des Problembereichs statt. *STEVE* kann sowohl durch Fragestellungen als auch durch aktives Eingreifen in den Ablauf zur Problemlösung beitragen. In einer Gruppe nimmt *STEVE* somit die Rolle eines virtuellen Teammitglieds ein.

Als virtuelles Teammitglied kann auch der virtuelle Tutor des ITS *HabiPro* angesehen werden, der dort als *Simulated Student* (dt.: simulierter Student) bezeichnet wird und einer Gruppe von Lernenden in verschiedenen Problemsituationen tutoriellen Support anbieten kann (vgl. Vizcaíno & Du Boulay, 2002, S.2f). Als Domäne von *HabiPro* wird die Java-Programmierung in virtuellen Teams genannt. Die Zusammensetzung der Gruppe findet bei diesem System jedoch keine weitere Beachtung. Als weitere relevante CSCL-Systeme mit tutoriellen Komponenten sind *COLER* zur Modellierung von Entity-Relationship-Diagrammen (vgl. Constantino-Gonz'alez & Suthers, 2001), das im Rahmen des Projekts *EPSILON* entwickelte *COMET* zur objektorientierten Modellierung (vgl. Soller, 2004) oder auch *Algebra-JAM* (vgl. Singley et al., 2000) zu nennen.

Diese Liste von Systemen, die einzelne Lernende, aber auch virtuelle Teams beim Lernen unterstützen, lässt sich noch weiter fortsetzen, jedoch findet sich derzeit kein ITS, das alle nachfolgenden Kriterien vollständig erfüllt und sich somit auf die eingangs beschriebene Problemsituation anwenden lässt:

1. Die Domäne des ITS umfasst die objektorientierte Programmierung in Java.
2. Das System ermöglicht die synchrone kollaborative Problemlösung.
3. Die tutorielle Komponente liefert jeder Gruppe adäquaten Support auf der Basis eines geeigneten Rollenmodells.

Das Ziel der vorliegenden Arbeit besteht darin, einen Beitrag zur Schließung dieser Forschungslücke zu liefern.

1.3 Forschungsfragen

Aus der skizzierten Situation heraus entstand 2001 das Projekt *VitaminL* (Virtuelle Teams: Analyse und Modellierung in netzbasierten Lernumgebungen) mit dem Ziel, für die Domäne der objektorientierten Programmierung in Java ein CSCL-System bereitzustellen, dessen tutorielle Komponente abhängig von der Teamzusammensetzung angepassten Support bietet, so dass weder die Motivation der Lernenden durch zuwenig Hilfe noch der beabsichtigte Lerneffekt durch zuviel Unterstützung

beeinträchtigt werden. Das im Rahmen des Projekts *VitaminL* geplante Gesamtvorhaben wirft einige Fragen auf, die sowohl für das Projekt als auch für das Forschungsgebiet CSCL von Interesse sind.

> Lassen sich optimale Gruppenzusammensetzungen für die synchrone Java-Programmierung in virtuellen Teams festlegen?
>
> Welche Probleme treten während der gemeinsamen, synchronen Bearbeitung von Übungsaufgaben aus der Domäne der objektorientierten Java-Programmierung in einem CSCL-System auf?
>
> Wie lassen sich die Probleme kategorisieren?
>
> Wie können Problemsituationen zuverlässig erkannt werden?
>
> Welche Probleme lassen sich generell erkennen?
>
> Welchen Kategorien gehören die erkannten beziehungsweise erkennbaren Probleme an?
>
> Wie sieht geeigneter tutorieller Support für die erkannten Probleme in Abhängigkeit von der Teamzusammensetzung aus?
>
> Wie kann die Zusammensetzung einer Gruppe bezüglich eines Rollenmodells erkannt werden, während ihre Mitglieder in einem CSCL-System gemeinsam tätig sind?

Die vorliegende Arbeit setzt sich primär mit Fragestellungen zur Erkennung von Rollenzugehörigkeiten einzelner Mitglieder sowie zur Identifikation von Problemsituationen während der Zusammenarbeit auseinander. Die Ziele liegen einerseits in der Entwicklung von geeigneten Verfahren zur Erkennung von Rollenausprägungen und andererseits in der Ausarbeitung eines Konzepts für ein Verfahren zur Problemerkennung. Dabei sind Umsetzung in Form von Software-Komponenten als Bestandteile einer Tutorkomponente sowie deren Integration in eine geeignete Lernumgebung zu berücksichtigen. Innerhalb der Tutorkomponente werden Analyseergebnisse an nachgelagerte Komponenten weitergereicht, wo sie ausgewertet und zur Generierung von adäquatem Support herangezogen werden können.

1.4 Methodisches Vorgehen

Die im Rahmen der vorliegenden Arbeit durchgeführten Untersuchungen basieren auf einer umfangreichen Wissensbasis, deren einzelne Elemente mittels einer Vielzahl von Beobachtungen und Befragungen ermittelt wurden.

1.4.1 Beobachtungen

Beobachtungen gelten als wesentliches Hilfsmittel zur Informationsgewinnung im Rahmen wissenschaftlicher Untersuchungen (s. Kap.2.4.3.2), auf das auch in der

vorliegenden Arbeit nicht verzichtet werden kann. So wurden beispielsweise im Vorfeld des VitaminL-Projekts die Teilnehmer einer Einführungsveranstaltung zur Programmierung in Java von anwesenden Tutoren bei ihren ersten Programmierversuchen beobachtet und aufgetretene Fehler protokolliert mit dem Ziel, erste Erkenntnisse über typische Anfängerfehler zu erhalten (s. Kap.7.2).

1.4.2 Benutzertests

Als spezielle Form der Beobachtung können die Benutzertests verstanden werden, die ein wesentliches Element der innerhalb dieser Arbeit verwendeten Methoden darstellen. Sie dienen – in Zusammenhang mit anschließend auszufüllenden Fragebögen – der Erprobung und Evaluierung von bestimmten Konzepten. Insbesondere wird anhand einer Machbarkeitsstudie die prinzipielle Durchführbarkeit des geplanten Vorgehens und damit einhergehend auch die Akzeptanz eines Systems zur verteilten, synchronen Java-Programmierung überprüft (s. Kap.8.1). Auch die Überprüfung neuer oder überarbeiteter Komponenten und Werkzeuge innerhalb des Systems wird mit Benutzertests durchgeführt. Dies betrifft insbesondere die Dialogkomponente des VitaminL-Systems (s. Kap.8.2) sowie den Gruppeneditor (s. Kap.8.3). Darüber hinaus wurden sämtliche Benutzertests durch das VitaminL-System mitprotokolliert. Das Ergebnis sind Protokolldateien, die sämtliche Benutzeraktionen je eines Benutzertests enthalten. Die prinzipielle Strukturierung der Benutzertests ist in Kapitel 8.1.2 beschrieben.

1.4.3 Analyse von Protokolldateien

Die während eines Benutzertests vom System erfassten Interaktionen der Teilnehmer mit der VitaminL-Applikation werden auf dem VitaminL-Server protokolliert und in einem XML-Format in einer dem Benutzertest zugeordneten Protokolldatei abgespeichert (s. Kap.8.4.4). Diese Protokolldateien bilden, zusammen mit den Ergebnissen eines Rollenfragebogens, die Grundlage für Untersuchungen zu Rollenzugehörigkeiten der Teilnehmer (s. Kap.10). Ferner wurden sie einer mit intellektuellem Aufwand durchgeführten Analyse zu Problemsituationen und deren Erkennbarkeit zugeführt (s. Kap.11).

1.4.4 Fragebögen

Neben Beobachtungen stellen Befragungen ein weiteres wichtiges Instrument für die Informationsgewinnung wissenschaftlicher Untersuchungen dar (s. Kap.2.4.3.3). Im Rahmen der vorliegenden Arbeit wurden Befragungen anhand diverser Fragebögen durchgeführt. Speziell für die Rollenanalyse (s. Kap.10) wurde der Fragebogen von Spencer & Pruss (1995) (s. Kap.2.5.5; Spencer & Pruss, 1995, S.75) elektronisch umgesetzt und den Teilnehmern von Benutzertests über eine

Web-Seite zur Verfügung gestellt. Die erhaltenen Informationen werden zusammen mit den zugehörigenen Protokolldateien der jeweiligen Teilnehmer in der Rollenanalyse mittels statistischer Verfahren ausgewertet. Des weiteren wurden Fragebögen im Anschluss an Benutzertests von deren Teilnehmern beantwortet, um bestimmte Aspekte der Benutzertests zu erfragen beziehungsweise zu evaluieren. Auf diese Weise konnten beispielsweise Basiskonzepte des VitaminL-Systems, wie die Dialogkomponente des VitaminL-Systems (s. Kap.8.2) oder der Gruppeneditor (s. Kap.8.3) , überprüft und anhand der Fragebogenergebnisse stetig verbessert werden.

1.4.5 Statistische Verfahren

Die Rollenanalyse (s. Kap.10) befasst sich mit der Fragestellung, ob statistisch nachweisbare lineare Zusammenhänge zwischen den Rollenausprägungen von Teilnehmern und deren Verhalten bei der synchronen Java-Programmierung in virtuellen Teams existieren. Dazu kommen neben der Bestimmung des linearen Zusammenhangs mittels Maßkorrelationskoeffizient und Bestimmtheitsmaß (s. Kap.10.1.2.2) auch die Durchführung einer Korrelationsanalyse (s. Kap.10.1.2.3) und einer Regressionsanalyse (s. Kap.10.1.2.4) zur Anwendung. Die einzelnen statistischen Verfahren werden in Kapitel 10.1.2 eingeführt.

1.4.6 Iterative Software-Entwicklung

Die im Rahmen des VitaminL-Projekts entwickelte Software wurde über einen längeren Zeitraum in mehreren Schritten entwickelt. Ausgehend von einem Prototypen zur Überprüfung der generellen Durchführbarkeit des geplanten Projekts (s. Kap.8.1) wurde das System sukzessive um neue Komponenten ergänzt, mit denen jeweils spezielle Bedienkonzepte realisiert wurden. Diese wurden stets mit Benutzertests und entsprechenden Fragebögen evaluiert und führten – wie beispielsweise im Fall des Gruppeneditors oder auch der Dialogschnittstelle – zu überarbeiteten, verbesserten Versionen der entsprechenden Komponenten. Auch die Integration der Tutorkomponente in das VitaminL-System erfolgt gemäß dem Ansatz der iterativen Software-Entwicklung, so dass jeder Entwicklungsschritt ein stabiles Software-System lieferte.

Mit den in diesem Kapitel vorgestellten Instrumenten ergibt sich ein vielschichtiger Methodenmix für die in dieser Arbeit durchgeführten Untersuchungen.

1.5 Nutzen für die Praxis

Diese Arbeit versteht sich als Beitrag zu einer neuartigen Form von Lernumgebung, die ihre primäre Anwendung in kurzzeitigen, synchronen Kollaborationen virtueller Teams im Bereich der objektorientierten Software-Entwicklung mit der

Programmiersprache Java sieht. Das der vorliegenden Arbeit zugrundeliegende CSCL-System wird bereits erfolgreich zur Unterstützung der o.g. Lehrveranstaltungen eingesetzt: Dozenten können ausgewählte Quelltexte über das System an die Arbeitsplätze von Studierenden übertragen und somit den theoretischen Lehrstoff anhand von Beispielen verdeutlichen, Studierende nutzen das System zur verteilten, gemeinsamen Bearbeitung von Übungs- und Hausaufgaben sowie von Semesterarbeiten.

Eine in der Tutorkomponente enthaltene Analysekomponente beobachtet jedes Team während dessen Zusammenarbeit und generiert Informationen über die Zusammensetzung des Teams auf der Basis eines geeigneten Rollenmodells sowie weitere für den Support relevante Kennzahlen. Anhand dieser Daten ist die tutorielle Komponente in der Lage, in erkannten Problemsituationen Supportangebote zu erzeugen, die an die Bedürfnisse des Teams angepasst sind – und dies bei einer im Vergleich zu einem realen Tutor wesentlich höheren Verfügbarkeit unabhängig von Ort und Zeit. Jedem Team wird somit ein individueller virtueller Tutor als weiteres Teammitglied zur Seite gestellt.

In der Folge kann eine Entlastung realer Tutoren von einer Vielzahl bekannter Problemsituationen erwartet werden, da diese Probleme nun weitgehend vom CSCL-System und seiner Tutorkomponente behandelt werden. Die auf diese Weise freigesetzten Kapazitäten können für die Bearbeitung anspruchsvollerer Probleme wie beispielsweise Fragen objektorientierten Designs genutzt werden.

1.6 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in einen theoretischen und einen empirischen Teil. Im Theorieteil werden wichtige Begriffe definiert sowie Sachverhalte und Methoden erläutert, die für das Verständnis dieser Arbeit unabdingbar sind. Der empirische Teil umfasst die im Rahmen der vorliegenden Arbeit durchgeführten Untersuchungen einschließlich zugehöriger Evaluationen, Ergebnisse und Folgerungen.

Zunächst findet in Kapitel 2 die begriffliche Heranführung an Gruppen und Teams als soziale Organisationseinheiten statt, die als solche eine unverzichtbare Grundlage für alle nachfolgenden Betrachtungen darstellen. Ergänzt werden diese Begriffe um Erläuterungen zu Gruppenprozessen und Gruppenstrukturen, welche letztlich zu den Rollenmodellen als eine Form der strukturellen Sichtweise von Gruppen führen.

Im Anschluss daran (Kapitel 3) folgt die Vorstellung typischer Dienste, die auf der Vernetzung von Computern unter Verwendung gängiger Informations- und Kommunikationstechnologien beruhen und diverse Ausprägungen computervermittelter Kommunikation zwischen den Computern beziehungsweise deren Benutzern mit sich bringen. Dies führt zu neuen Gruppenformen, sog. *virtuellen Gruppen*, die inklusive ihrer Vor- und Nachteile gegenüber konventionellen Gruppen zum Abschluss von Kapitel 3 diskutiert werden.

Die computer-basierte Unterstützung von Zusammenarbeit innerhalb virtueller Gruppen ist Gegenstand des Forschungsgebiets CSCW (*Computer Supported Collaborative Work*), welches in Kapitel 4 erläutert wird. Neben der Historie von CSCW, beteiligten Forschungsdisziplinen und -inhalten sowie relevanten Begriffen werden insbesondere spezifische Werkzeuge zur Unterstützung der Zusammenarbeit vorgestellt.

In Kapitel 5 schließt sich die Vorstellung des computer-gestützten Lernens in Gruppen (CSCL; *Computer Supported Collaborative Learning*) an, wobei dessen Einordnung in die Forschungslandschaft einerseits sowie das Gebiet des E-Learning andererseits behandelt werden. Neben didaktischen Konzepten werden auch hier spezifische Werkzeuge zur Unterstützung kollaborativen Lernens präsentiert.

Die in Kapitel 6 betrachteten inteligenten Tutor-Systeme (ITS) stellen eine spezielle Form des E-Learning dar, mit deren Hilfe im Regelfall ein einzelner Lernender[1] innerhalb eines bestimmten Fachgebiets Unterstützung durch einen sogenannten *virtuellen Tutor* erhält. Solche ITS, die Gruppen von Lernenden unterstützen, werden auch als *Intelligent Cooperative/Collaborative Learning System* (ICLS) bezeichnet. Die Betrachtung umfasst neben typischen Konzepten der KI (*Künstliche Intelligenz*) auch ein Architekturmodell als Rahmen für die Ausgestaltung intelligenter tutorieller Systeme. Die Präsentation ausgewählter Systeme, die Einfluss auf die vorliegende Arbeit haben, rundet die Betrachtungen zu ITS ab. Damit endet der Theorieteil dieser Arbeit.

Der empirische Teil dieser Arbeit beginnt in Kapitel 7 mit einer Heranführung an das typische Lehr-/Lernszenario beim Erlernen einer Programmiersprache wie Java. Anhand einer empirischen Studie unter den Teilnehmern einer Einführungsveranstaltung zum Thema *Programmierung in Java* werden typische Fehler und ihre möglichen Ursachen aufgezeigt. Basierend auf der Idee, mittels eines neuartigen CSCL-Systems respektive eines ITS virtuelle Teams bei der synchronen Java-Programmierung zu unterstützen, lassen sich aus dieser Studie wichtige Forschungsfragen ableiten, die sich einerseits der Identifikation von Problemen während der synchronen Zusammenarbeit im virtuellen Team widmen und die andererseits tutorielle Unterstützungsangebote betrachten, die an die Bedürfnisse des jeweiligen Teams angepasst sind.

Technische Aspekte des geplanten Systems, der *VitaminL-IDE* werden im darauf folgenden Kapitel 8 behandelt. Dazu wird anhand einer Studie, die mit einem ersten Prototypen der VitaminL-IDE durchgeführt wurde, die prinzipielle Machbarkeit und Durchführbarkeit des geplanten Ansatzes aufgezeigt. Neben speziellen Konzepten für Kommunikation und Kooperation wird auch die Architektur des Systems als CSCL-System eingehend diskutiert. Am Ende von Kapitel 8 steht die Integration einer Tutorkomponente in die bestehende Architektur: Das VitaminL-System wird von einem CSCL-System zu einem ICLS erweitert.

Kapitel 9 beinhaltet die Festlegung des VitaminL-Arbeitsmodells. Dieses besteht aus dem Vorgehensmodell, welches die typische Vorgehensweise virtueller Teams

[1] Die Unterstützung von Lerngruppen durch ein ITS stellt bislang eher die Ausnahme dar.

beim Arbeiten mit dem VitaminL-System – strukturiert in Prozesse, Teilprozesse, Phasen und Schritte – erfasst. Es stellt somit eine Art organisatorischen Rahmen für die verteilte, synchrone Java-Programmierung in virtuellen Teams dar. Das Rollenmodell von VitaminL als weiterer Bestandteil des Arbeitsmodells charakterisiert die für eine erfolgreiche Zusammenarbeit benötigten Teamfunktionen. Die Rollen des Rollenmodells werden anschließend auf die einzelnen Einheiten des Vorgehensmodells abgebildet. Hierbei findet eine Gewichtung der Rollen nach deren jeweiliger Relevanz für das Vorgehensmodell statt. Das Resultat ist das VitaminL-Arbeitsmodell, welches die geplante Tutorkomponente maßgeblich mitgestaltet.

Im Anschluss wird in Kapitel 10 die Rollenanalyse durchgeführt. Hierunter wird die Entwicklung eines Verfahrens zur Bestimmung von Rollenausprägungen der Mitglieder eines virtuellen Teams auf der Grundlage des im vorigen Kapiel spezifizierten Rollenmodells verstanden. Dazu werden zunächst die in die Analyse eingehenden Sitzungsdaten charakterisiert und formal definiert, gefolgt von einer detaillierten Festlegung des für die Rollenanalyse eingesetzten Verfahrens. Es schließt sich eine Diskussion der Analyseergebnisse an. Darauf aufbauend folgt eine Evaluierung dieser Ergebnisse mittels eines in das VitaminL-System integrierten Prototypen einer Rollenanalysekomponente. Die Resultate dieser Evaluierung werden den ursprünglichen Analyseergebnissen gegenübergestellt. Eine Zusammenfassung, in der ausführlich das Weiterentwicklungspotential der Rollenanalyse diskutiert wird, schließt dieses Kapitel ab.

Die Konzeptstudie zur Erkennung von Problemsituationen während der Zusammenarbeit ist Betrachtungsgegenstand des 11. Kapitels. Basierend auf dem VitaminL-Vorgehensmodell wird zunächst eine Klassifikation derjenigen Probleme erarbeitet, die für den betrachteten Kontext relevant und möglich sind. In diese Problemklassifikation fließen unter anderem die Ergebnisse der in Kapitel 7 durchgeführten Studie ein. Anhand weiterer Benutzertests werden für das behandelte Lernszenario typische Probleme erfasst und im Hinblick auf deren Erkennung durch eine Problemanalysekomponente als Teil der Tutorkomponente analysiert. Das Kapitel endet mit einer Diskussion der erzielten Ergebnisse, welche weiterführende Konzepte im Hinblick auf die genannte Problemanalysekomponente umfasst.

In Kapitel 12 werden die Resultate sowohl der Rollen- wie auch der Problemanalyse nochmals zusammengefasst und bewertet. Hierzu gehört auch ein Ausblick auf die weitere Entwicklung der Tutorkomponente, wobei auch die jeweiligen Evolutionspotentiale beider Analyseverfahren einzubeziehen sind.

Im Anhang finden sich vertiefende Angaben zu technischen Details der VitaminL-Software (Kapitel A), zu Rollenmodellen (Kapitel B), zu Rollenprofilen der Teilnehmer (Kapitel C), zur Rollenanalyse (Kapitel D) sowie zur Problemanalyse (Kapitel E). Ein Glossar und das Literaturverzeichnis vervollständigen die Arbeit.

1.7 Verwendete Hilfsmittel

Neben den im Literaturverzeichnis aufgeführten Quellen kamen sowohl bei der Erstellung dieser Arbeit als auch im Rahmen des VitaminL-Projekts weitere Hilfsmittel in Form von Software-Produkten zum Einsatz:

1. *VitaminL-Software*:
   Die VitaminL-Software – und ihre Weiterentwicklung zu einem ICLS – stellt nicht nur den Betrachtungsgegenstand der vorliegende Arbeit dar, sie ist als Arbeitsplattform in Benutzertests und zur Gewinnung von Sitzungsdaten (in Form von XML-Protokolldateien) auch das wohl wichtigste Hilfsmittel der vorliegenden Untersuchungen.

2. *Java*:
   Die Programmiersprache *Java* liefert einerseits den Kontext für diese Arbeit und ist andererseits auch die Programmiersprache, in welcher die gesamte VitaminL-Software geschrieben ist. Alle dafür notwendigen Entwicklungswerkzeuge (Java-Compiler, Java-Interpreter etc.) werden in einem Software-Paket, dem sogenannten JDK (*Java Development Kit*; dt.: Java-Entwicklungssystem) von der Firma *Sun microsystems* kostenlos auf deren Web-Server[2] zur Verfügung gestellt.

3. *eclipse*:
   Bei *eclipse*[3] handelt es sich um ein *Open-Source-Framework*, das heißt eine Anwendungsarchitektur auf der Basis frei verfügbarer Quelltexte, die mittels sogenannter *Plugins* vielfältig erweiterbar ist. Auf diese Weise lässt sich ein breites Anwendungsspektrum abdecken, das sich von der integrierten Entwicklungsumgebung (IDE; *Integrated Development Environment*) für Programmiersprachen wie Java bis hin zur Plattform für die Entwicklung von *Rich-Client-Applications* erstreckt. eclipse selbst ist vollständig in Java geschrieben und wurde im VitaminL-Projekt hauptsächlich als Java-IDE verwendet, kam aber auch als *Front-end* des auf einem Linux-Server laufenden Versionsverwaltungssystems glosscvs sowie zur Administration einer ebenfalls auf einem Linux-Server laufenden *MySQL*-Datenbank zum Einsatz.

4. *Linux*:
   Das freie Betriebssystem kam im Rahmen des VitaminL-Projekt auf mehreren Servern zum Einsatz. Somit konnten unterschiedliche Dienste vom einfachen Web-Server des VitaminL-Projekts über einen CVS-Server zur Verwaltung der VitaminL-Quelltexte bis hin zum VitaminL-Server selbst auf einer stabilen Grundlage angeboten werden.

5. *MySQL*:
   MySQL[4] steht als freie Software unter *General Public License* und ist ein rela-

[2] `http://java.sun.com/` - letzter Zugriff am 11.12.2007
[3] `http://www.eclipse.org/` - letzter Zugriff am 11.12.2007
[4] `http://mysql.com/` - letzter Zugriff am 11.12.2007

tionales Datenbankverwaltungssystem (DBMS; *Database Management System*). Im VitaminL-Projekt läuft eine MySQL-Datenbank auf dem VitaminL-Server, die einerseits für die Benutzerverwaltung zuständig ist und andererseits die Wissensbasis für die in der vorliegenden Arbeit behandelten Rollenanalyse enthält.

6. *OpenOffice.org Calc*:
   Die Tabellenkalkulation des freien Office-Pakets[5] kam während der Rollenanalyse zur Aufbereitung der Analyseergebnisse sowie zur Generierung von Diagrammen zum Einsatz.

7. *LaTeX*:
   Die vorliegende Arbeit wurde komplett mit LaTeX, dem freien Textsatzsystem, erstellt. Es kam dabei die *Miktex*[6]-Distribution zum Einsatz.

8. *Gimp*:
   Mit der freien Bildverarbeitungssoftware *Gimp*[7] wurde eine Vielzahl der Graphiken dieser Arbeit erstellt.

9. *Literarix*:
   Sämtliche in dieser Arbeit verwendete Literatur wurde mit dem Literaturverwaltungsprogramm *Literarix*[8] verwaltet. Ein Export in das BibTeX-Format erlaubt das nahtlose Zusammenspiel mit LaTeX.

Des Weiteren wurden einige kleinere Hilfsprogramme in Java entwickelt und eingesetzt. Dazu zählen der *Logfile-Viewer*, mit dessen Hilfe aufgezeichnete Benutzertests eingelesen und – ohne menschliche Teilnehmer – wiederholt werden können. Ferner wurde eine Java-Applikaton, das *VDBTool* entwickelt, mit dem sowohl XML-Protokolldateien und Rollenprofile von Teilnehmern in eine Wissensbasis (realisiert als MySQL-Datenbank) einpflegt werden können als auch wesentliche Berechnungen der Rollenanalyse unter Verwendung dieser Wissensbasis durchgeführt werden können. Weitere nennenswerte Hilfsmittel wurden nicht verwendet.

1.8 Anmerkungen zur Sprache

Beim Verfassen der Arbeit wurden folgende Gesichtspunkte festgelegt, die beim Lesen der Arbeit ebenso zu berücksichtigen sind:

- Wenn in der Arbeit beispielsweise von *dem Studierenden* die Rede ist, so soll dies männliche wie weibliche Studierende gleichermaßen ansprechen. Zu Gunsten einer flüssigen und leicht lesbaren Arbeit wurde auf die zusätzliche

[5] `http://de.openoffice.org/` - letzter Zugriff am 11.12.2007
[6] `http://miktex.org/` - letzter Zugriff am 11.12.2007
[7] `http://www.gimp.org/` - letzter Zugriff am 11.12.2007
[8] `http://www.literarix.de/` - letzter Zugriff am 11.12.2007

Nennung auch der weiblichen Form – wie beispielsweise *der Student* beziehungsweise *die Studentin* oder *der/die Studierende* – verzichtet. Eine geschlechterspezifische Ausgrenzung aufgrund der überwiegenden Verwendung der männlichen Form ist vom Autor dieser Arbeit zu keinem Zeitpunkt beabsichtigt.

- Anglizismen werden verhältnismäßig sparsam verwendet, kommen jedoch immer dann zum Einsatz, wenn die englischen Begriffe etabliert sind und eine Eindeutschung daher nicht sinnvoll ist.

- Zitate, insbesondere Kommunikationsbeiträge aus Benutzertests, werden ohne Korrekturen oder Anpassungen an gültige Rechtschreibregeln verwendet.

- Englische Zitate werden stets im Original belassen.

- Sämtliche an den empirischen Studien teilnehmenden Personen sind anonymisiert und werden mit T_i (mit $i \in N$) bezeichnet. Um die Anonymität vollständig gewährleisten zu können, mussten – abweichend von der ansonsten unveränderten Verwendung von Zitaten – an einigen wenigen Stellen Kommunikationsbeiträge geändert und Namen durch das jeweilige T_i ersetzt werden.

- Verweise auf Tabellen, Abbildungen, Kapitel, Seitenzahlen und ähnliches werden im Text ausgeschrieben, in Anmerkungen wie Literaturverweisen und Fußnoten jedoch aus Gründen der besseren Lesbarkeit abgekürzt (*Tab.*, *Abb.*, *Kap.*, *S.* etc.).

Diese Regeln sollen eine gewisse Transparenz hinsichtlich der Schreibweise erzielen.

Kapitel 2

Gruppen: Prozesse und Strukturen

In einer Arbeit, die das kollaborative Lernen zum Forschungsgegenstand hat, müssen auch Gruppen als Grundlage dieser speziellen Lernform angemessen berücksichtigt werden. Zur Heranführung an Gruppen und damit verbundene, für die vorliegende Arbeit relevante Aspekte dient dieses Kapitel: Nach einer intuitiven, historisch motivierten Einleitung in diesen Themenkomplex folgen zunächst einige begriffliche Definitionen. Es schließen sich Betrachtungen zu Gruppenprozessen und -strukturen an, die letztlich zu den Rollenmodellen führen. Rollenmodelle stellen einen bewährten Ansatz dar, den strukturellen Aufbau von Gruppen zu beschreiben, und finden auch in der vorliegenden Arbeit Verwendung als Ausgangspunkt für die Generierung angemessener tutorieller Unterstützung von Lerngruppen.

„Gruppen sind ein elementarer Bestandteil menschlichen Zusammenlebens" (Haake et al., 2004, S.42). So definierte bereits Aristoteles[1] den Menschen als *Zoon politicon*, also als ein Wesen, das darauf angewiesen ist, Gesellschaften zu bilden (vgl. Kriz & Nöbauer, 2002, S.14), da nur die Gruppe in der Lage ist, dem Einzelnen Nahrung, Schutz und Unterkunft in ausreichendem Maße auch unter widrigen Umständen zu bieten (vgl. Mann, 2001, S.50f; Kriz & Nöbauer, 2002, S.14f; Spencer & Pruss, 1995, S.11ff).

In der modernen Arbeitswelt ist die Gruppe in Form von Arbeitsgruppen oder Teams als Organisationseinheit innerhalb von Unternehmen nicht mehr wegzudenken. Als Gründe für die Verbreitung von Teams in Organisationen *„werden in erster Linie (1) die Veränderung der Märkte und damit der Wettbewerbssituation sowie (2) die Einführung neuer Technologien durch den Siegeszug der Mikroelektronik angesehen ... Zudem wird immer wieder der (3) Wertewandel in der Diskussion zur (Wieder-)Belebung der Gruppenarbeit angeführt"* (Kauffeld, 2001, S.7).

Aber auch dem Lernen in der Gruppe wird mittlerweile eine hohe Bedeutung für den Wissenserwerb zugewiesen: *„Das Lernen in der Gruppe erleichtert die Ausei-*

[1] gr. Philosoph, * 384 v. Chr. in Stageira/Makedonien, †322 v. Chr. in Chalkis/Euböa

nandersetzung mit dem Lerngegenstand aus unterschiedlichen Blickwinkeln. Diskussionen in geeigneten Lerngruppen helfen dabei, den Stoff tiefgreifend zu verstehen. Darüber hinaus wird das Verständnis des Lernstoffes dadurch gefördert, dass Lernende gegenüber ihren Mitlernern gelegentlich die Rolle von Lehrern einnehmen und Inhalte aus ihrer Sicht vermitteln" (Haake et al., 2004, S.80). Insofern erscheint es durchaus sinnvoll, diese spezielle Lernform zu unterstützen und zu förden. Dazu ist es aber zunächst notwendig, zu klären, was eine Gruppe überhaupt ist.

2.1 Begriffsklärung

Eine Arbeit, die sich mit dem Arbeiten und Lernen in der Gruppe befasst, sollte ihrem Leser klare Begriffe und präzise Definitionen dazu anbieten können. Unglücklicherweise existiert in der meist sozialpsychologischen Literatur eine Vielzahl unterschiedlicher Definitionen zum Begriff der Gruppe.

2.1.1 Gruppen

Als eine der wohl einfachsten Definitionen des Gruppenbegriffs kann die folgende Definition nach Lindgren angesehen werden: *„Wenn zwei oder mehr Personen in irgendeiner Beziehung zueinander stehen, bilden sie eine Gruppe"* (Lindgren, 1973, S.347 in Sader, 2002, S.37). Diese Definition ist jedoch in ihrer Form derart allgemeingültig und universell anwendbar, dass auch soziale Aggregate, also bloße Ansammlungen von Individuen mit gemeinsamen Merkmalen wie Arbeitslose, Schüler oder Wartende in einer Warteschlange bereits als Gruppe angesehen werden können.

Eine erste Einschränkung liefern Zimbardo & Gerrig in ihrer Definition: *„Von einer Gruppe sprechen wir dann, wenn zwischen zwei oder mehr Personen eine Interaktion stattfindet, bei der jeder den/die anderen beeinflusst und von dem/den anderen beeinflusst wird"* (Zimbardo & Gerrig, 2000, S.723). Das Vorhandensein von Interaktion kann als Mindestanforderung angesehen werden, um in einem ersten Schritt Gruppen von sozialen Aggregaten zu unterscheiden. Es wird üblicherweise erst dann von Gruppen gesprochen, *„wenn über eine gewisse Dauer Interaktionen zwischen den Personen entstehen. Diese Unmittelbarkeit der Beziehung, die direkten Interaktionen können als Mindestanforderung an jegliche Form von Gruppe ... betrachtet werden"* (vgl. Kriz & Nöbauer, 2002, S.16). Ähnliches findet sich auch in der nächsten Definition: *„For a collection of individuals to be considered a group there must be some interaction"*(Hare, 1962, S.10). Folglich kann noch nicht von einer Gruppe gesprochen werden, wenn zwei oder mehr Individuen sich im selben Raum aufhalten, solange keine Interaktion zwischen diesen beiden stattfindet.

Eunson betrachtet die Gruppe als einen *„Mechanismus, der das Individuum mit Organisationen und mit der Gesellschaft als Ganzes verbindet"* (Eunson, 1990,

S.425). Zwecks Abgrenzung von Gruppen zu sozialen Aggregaten, die der Autor als *soziale Kategorien* oder *Gesamtheit* bezeichnet (vgl. EUNSON, 1990, S.426), stellt er – neben der bereits erwähnten Interaktion – eine weitere Anforderung an eine Gruppe: *„Mitglieder einer Gruppe wirken zusammen, um gemeinschaftliche Ziele zu erreichen"* (EUNSON, 1990, S.427).

Auch BALES versucht, die Gruppe von sozialen Aggregaten und Gesamtheiten abzugrenzen, und prägt mit seiner Definition den Begriff der Kleingruppe: *„A small group is defined as any number of persons engaged in interaction with each other in a single face-to-face meeting or a series of meetings, in which each member receives some impression or perception of each other member distinct enough so that he can, either at the time or in later questioning, give some reaction to each of the others as an individual person, even though it be only to recall that the other was present"*(BALES, 1950, S.33). Wichtige Aspekte dieser Definition sind neben der bereits zitierten Interaktion auch die Nachhaltigkeit der Begegnung zwischen den beteiligten Personen, bewirkt durch wiederholte und/oder längerfristige Treffen, sowie die Direktheit und Unmittelbarkeit dieser Treffen (*face-to-face*). Die Forderung nach der Unmittelbarkeit der Interaktion zwischen den Beteiligten muss jedoch zwischenzeitlich – bedingt durch die rasanten Fortschritte insbesondere der letzten Jahre in den Informations- und Kommunikationstechnologien – relativiert werden, so dass der Gruppenbegriff in der vorliegenden Arbeit auf der Definition von BALES basiert.

Es lassen sich zweifelsfrei weitere Definitionen des Gruppenbegriffs finden (vgl. KAUFFELD, 2001, S.11f; SADER, 2002, S.37f; ROSEMANN & BIELSKI, 2001, S.145ff), jedoch würde eine umfassende Aufzählung den Umfang dieser Arbeit bei weitem sprengen, ohne diese voranzubringen. Stattdessen werden im Folgenden Gruppen anhand der in DÖRING (2003, S.492) genannten vier Hauptmerkmale definiert (vgl. HAAKE ET AL., 2004, S.42):

- Ständige Kommunikation und Kommunikationsmöglichkeit,
- Abgrenzung von der Umwelt und innere Strukturierung der Gruppe,
- Zusammengehörigkeitsgefühl innerhalb der Gruppe sowie
- Zusammenarbeit und wechselseitige Unterstützung.

Dieses deckt sich im wesentlichen mit den Anforderungen, die in SADER (2002, S.39) oder KRIZ & NÖBAUER (2002, S.16f) an eine Gruppe beziehungsweise an eine Kleingruppe gestellt werden. So fasst auch HARE in ähnlicher Form zusammen: *„There are then, in sum, five characteristics which differentiate the* group *from a* collection of individuals. *The members of the group are in* interaction *with one another. They share a common* goal *and set of* norms*, which give direction and limits to their activity. They also develop a set of* roles *and a* network of interpersonal attraction*, which serve to differentiate them from other groups"* (HARE, 1962, S.10). Insbesondere der von HARE erwähnte Begriff der *Rolle* als Gruppencharakteristikum hat für die vorliegende Arbeit eine besondere Relevanz und wird

daher zu einem etwas späteren Zeitpunkt in Kapitel 2.5 wieder aufgegriffen und näher erläutert.

2.1.2 Gruppentypen

Wie schon anhand der Erwähnung des Begriffs der *Kleingruppe* ersichtlich ist, lassen sich Gruppen – unabhängig von der verwendeten Begriffsdefinition – durch die Bildung von Gruppentypen weiter strukturieren. Die Größe einer Gruppe zur Unterscheidung von Klein- und Großgruppen bildet dabei lediglich eines der für solch eine Strukturierung relevanten Merkmale. Nach Döring (2003, S.490ff) kann eine vollständige Strukturierung von Gruppen in Gruppentypen anhand folgender drei Merkmale erfolgen:

- Funktion der Gruppe: formale vs. informelle Gruppe
- Größe der Gruppe: Kleingruppe vs. Großgruppe
- Subjektive Bedeutung der Gruppe: Primärgruppe vs. Sekundärgruppe

2.1.2.1 Funktion der Gruppe

Ein wesentliches Strukturierungsmerkmal stellt die Funktion einer Gruppe dar, mittels welcher formale Gruppen von informellen Gruppen unterschieden werden. Formale Gruppen sind *„Gruppen, die zu einem bestimmten Zweck explizit gebildet werden"* (Haake et al., 2004, S.43) beziehungsweise die *„in erster Linie sachlich-instrumentellen Zielen dienen"*(Döring, 2003, S.491), wohingegen informelle Gruppen *„eher selbst organisiert und unter sozio-emotionalen Gesichtspunkten entstehen"* (Döring, 2003, S.491). Informelle Gruppen existieren oftmals parallel zu formalen Gruppen und davon weitestgehend unabhängig (vgl. Rosemann & Bielski, 2001, S.147f).

In der vorliegende Arbeit werden ausschließlich formale Gruppen in Form von Arbeitsgruppen beziehungsweise Lerngruppen betrachtet, informelle Gruppen sind aufgrund des gegebenen Kontexts (vgl. Kapitel 1.1) nicht relevant.

2.1.2.2 Größe der Gruppe

Die Einteilung anhand der Gruppengröße lässt eine Differenzierung in Kleingruppen und Großgruppen zu. Problematisch ist hierbei jedoch das Fehlen einer konkreten Personenzahl, anhand derer die Unterteilung vorzunehmen ist: *„There is also no definite cutting point between the small, intimate, face-to-face group and the large, formal group"* (Hare, 1962, S.9). In der Literatur finden sich daher durchaus unterschiedliche Angaben von fünf bis 30 Mitgliedern. Nach Sader ist es durchaus *„üblich, Dyaden (2 Personen), Kleinstgruppen (etwa 2 - 6 Personen), Gruppen (3 - etwa 30 Personen) und Großgruppen (zumeist über 25 Personen)*

zu unterscheiden" (SADER, 2002, S.39). Ähnlich unscharf formuliert es BALES: *„Groups of these kinds, ranging in number of persons involved from two to something around twenty, then, may be classed together als small groups on the basis of their amenability to study by a certain body of research procedures"* (BALES, 1950, S.i) Aufgrund dieser Unschärfe erfolgt die Einteilung in Klein- und Großgruppen üblicherweise nicht direkt anhand der Personenzahl, sondern indirekt anhand des Interaktionsgrads zwischen den einzelnen Mitgliedern, wobei angenommen wird, dass *„in Kleingruppen typischerweise alle Mitglieder regelmäßig miteinander interagieren"* (DÖRING, 2003, S.491). Dieses Merkmal findet sich auch in der bereits zitierten Definition von Kleingruppen nach BALES wieder (vgl. BALES, 1950, S.33).

Kleingruppen im Rahmen dieser Arbeit bestehen aus zwei bis fünf Mitgliedern (und sind nach SADER (2002) als Kleinstgruppen zu bezeichnen), wobei solche mit drei Mitgliedern die Mehrheit darstellen.

2.1.2.3 Subjektive Bedeutung der Gruppe

Wenn zwischen den einzelnen Mitgliedern einer Gruppe eine hohe sozioemotionale Bindung besteht, nennt man diese Gruppe auch *Primärgruppe*. Gruppen, deren Mitglieder untereinander keinen so ausgeprägten Bindungsgrad vorweisen, werden hingegen als *Sekundärgruppen* bezeichnet (vgl. HAAKE ET AL., 2004, S.42f). Klassische Beispiele für Primärgruppen sind Freundeskreis und Familie, wohingegen Lern- und Arbeitsgruppen in der Regel den Sekundärgruppen zuzuordnen sind. Dabei ist jedoch die Subjektivität der Bedeutung einer Gruppe für den Einzelnen zu berücksichtigen, das heißt es entscheidet letztlich jedes Mitglied einer Gruppe für sich, ob die Gruppe für ihn Primärgruppe oder Sekundärgruppe ist.

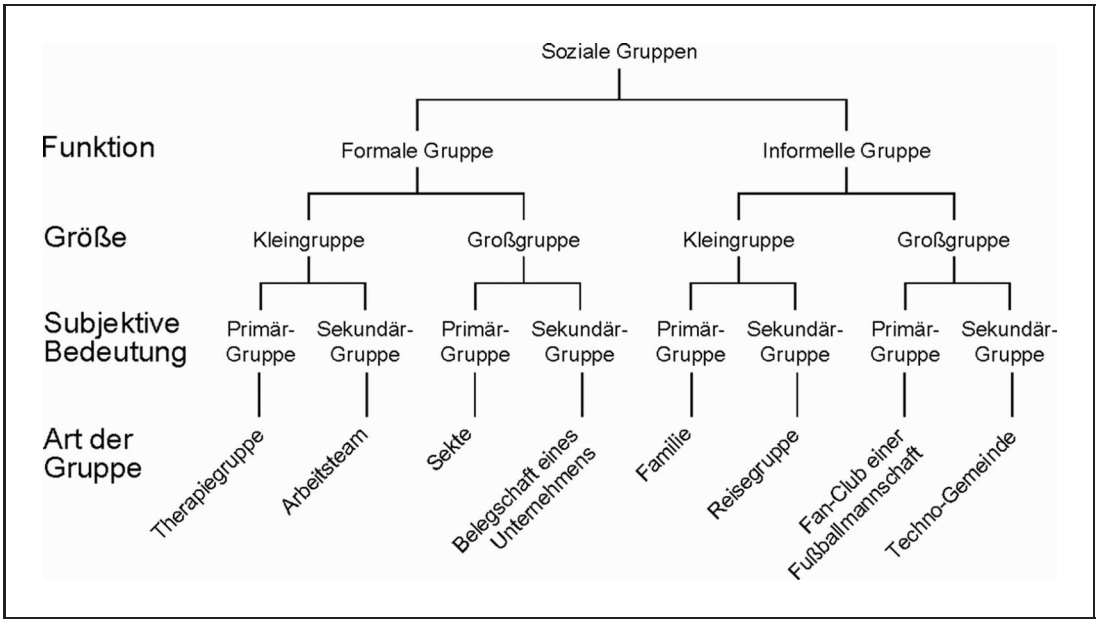

Abb. 2.1: Gruppentypen
(nach DÖRING, 2003, S.490)

Weitere etablierte Klassifikationsansätze zur Bildung von Gruppentypen finden sich beispielsweise in KAUFFELD (2001, S.17ff).

Während der gesamten Dauer des hier betrachteten Forschungsprojekts konnte kein einziger Fall beobachtet werden, in dem ein Teilnehmer einer Gruppe sich derart engagierte beziehungsweise so hohen sozialen Rückhalt in der Gruppe fand, dass diese für ihn eine Primärgruppe darstellt. Alle untersuchten Gruppen können folglich als Sekundärgruppen bezeichnet werden.

Mit den bisher genannten Definitionen ist der Begriff der Gruppen zwar umfassend erläutert, jedoch bleibt der Sinn und Zweck der Gruppenbildung weitgehend offen. Im Kontext der vorliegenden Arbeit sind zwei Formen des Miteinanders in der Gruppe näher zu betrachten: das Arbeiten in der Gruppe und das Lernen in der Gruppe. Daher werden nachfolgend zunächst die Begriffe *Arbeitsgruppe* und *Lerngruppe* definiert.

2.1.3 Arbeitsgruppen

Die Allgemeingültigkeit der Gruppendefinition von ZIMBARDO & GERRIG nehmen TEUFEL ET AL. zum Anlass, die Definition zu verfeinern und so den Begriff der Arbeitsgruppe festzulegen: *„Eine Arbeitsgruppe ist eine Gruppe mit einer bestimmten Aufgabe"* (TEUFEL ET AL., 1995, S.10).

Der Begriff der in der Definition geforderten *Aufgabe* ist zunächst so allgemeingültig gefasst, dass die Definition der Arbeitsgruppe sich auch auf einen Gesangsverein oder eine Volleyball-Mannschaft anwenden lässt (also auf Gruppen, die zunächst wohl kaum jemand als Arbeitsgruppen bezeichnen würde, obwohl auch diese sich sicherlich aufgrund einer gemeinsamen Aufgabe gebildet haben). Unter Berücksichtigung des Begriffs der Arbeit beziehungsweise des Arbeitens lassen sich – neben den bereits genannten Gruppenmerkmalen – zwei weitere wichtige Eigenschaften von Arbeitsgruppen beobachten. Zum einen besteht *„das Ziel einer Arbeitsgruppe [...] darin, dass ein bestimmtes Produkt (im weitesten Sinne) effizient erstellt wird und daraus ein Gewinn für die Organisation entsteht"*(HAAKE ET AL., 2004, S.203). Andererseits erfolgt die Motivation der Mitglieder einer Arbeitsgruppe extrinsich, das heißt durch äußere Einflusse wie beispielsweise der Entlohnung.

Da die typischen Tätigkeiten in einem Gesangsverein beziehungsweise in einer Volleyball-Mannschaft weder ernsthaft als Arbeit oder Produkterstellung im eigentlichen Sinne bezeichnet werden können und auch die Motivation kaum in Form einer Entlohnung erfolgt, sondern eher intrinsischer Natur ist, mag die o.g. Definition einer Arbeitsgruppe nach TEUFEL ET AL. hinreichend geeignet sein.

2.1.4 Lerngruppen

Treffen sich die Mitglieder einer Gruppe mit dem Ziel zu lernen, so spricht man von einer Lerngruppe: *„Eine Lerngruppe ist eine Gruppe, deren Mitglieder das Ziel verfolgen, Wissen zu erwerben, also zu lernen"* (HAAKE ET AL., 2004, S.203). Die

Motivation ist – im Gegensatz zu extrinsisch motivierten Arbeitsgruppen – vorrangig intrinsisch: Es gilt, die gesteckten (Lern-)Ziele zu erreichen. Diese spezielle Form des Lernens in der Gruppe wird auch als *kooperatives Lernen* bezeichnet.

Um den gewünschten Erfolg sicherzustellen, werden an Lerngruppen weitere Anforderungen gestellt (vgl. Haake et al., 2004, S.203f):

- Positive Abhängigkeit: Den Mitgliedern der Lerngruppe ist bewusst, dass sie ihr Ziel nur gemeinsam als Gruppe erreichen können, indem sie ihre Tätigkeiten aufeinander abstimmen.
- Individuelle Zurechenbarkeit/Persönliche Verantwortlichkeit: Jedes Mitglied ist verantwortlich für die von ihm übernommenen Aufgaben. Im gemeinsam erbrachten Ergebnis ist jede individuelle Leistung sichtbar und dem jeweiligen Mitglied zurechenbar.
- Fördernde Interaktion: Die Gruppenmitglieder unterstützen sich gegenseitig, so dass das gemeinsame Ziel auch gemeinsam erreicht werden kann.
- Soziale Kompetenz: Gruppenmitglieder gehen respektvoll miteinander um, Konflikte werden konstruktiv gelöst und innerhalb der Gruppe bildet sich ein Klima des gegenseitigen Vertrauens.
- Reflexion der Gruppenarbeit: Die Zusammenarbeit wird zum Gegenstand von Besprechungen innerhalb der Gruppe. Gleichzeitig erhalten die einzelnen Mitglieder Rückmeldungen zu ihren Beiträgen.

Mit diesen Merkmalen wird ein Rahmen für erfolgreiches kooperatives Lernen geschaffen.

Lerngruppen können nach ihrer Dauer in drei Arten unterschieden werden:

- Informelle kooperative Lerngruppen bilden sich spontan, ihre Lebensdauer reicht von wenigen Minuten bis hin zu einer Unterrichtsstunde, wobei sich der Fokus auf ein einziges Thema richtet.
- Formale kooperative Lerngruppen werden für die Dauer von einer oder mehreren Unterrichtsstunden gebildet und erarbeiten gemeinsam eine bestimmte Unterrichtseinheit. Der Lehrende gibt den organisatorischen Rahmen vor und steht für weitere Hilfestellungen zur Verfügung. Abschließend erfolgt eine Bewertung der Gruppenarbeit.
- Kooperative Basisgruppen besitzen eine Lebensdauer von mindestens einem Semester und zeichnen sich durch umfassende gegenseitige Unterstützung ihrer Mitglieder aus.

Neben der Lebensdauer und der Größe ist auch die Zusammensetzung einer Lerngruppe für ein (erfolgreiches) kooperatives Lernen relevant. Insbesondere die Faktoren *Größe* und *Zusammensetzung* können sich sowohl positiv als auch negativ

auf den Lernprozess auswirken: Mit zunehmender Gruppengröße nimmt der Anteil des einzelnen Mitglieds an der Gruppenarbeit ab, wohingegen in zu kleinen Gruppen nur geringe Meinungs- und Erfahrungsvielfalt zu finden ist. Doch gerade die Reichhaltigkeit gemeinsamen Wissens (und dazu zählt auch gemeinsam erarbeitetes Wissen) wird als einer der Vorteile kooperativen Lernens betont (vgl. Konrad & Traub, 2005, S.180). Auch eine gewisse Heterogenität der Gruppenzusammensetzung bzgl. Alter, Erfahrung, Kenntnisstand und anderer sozialer Faktoren ist in kooperativen Lernsituationen als durchaus positiv zu bewerten, da aufgrund dieser Heterogenität unterschiedliche Zugänge zum Lerngegenstand existieren. Dennoch wird ein Mindestmaß an Homogenität gefordert, um die Funktionsfähigkeit der Gruppe zu gewährleisten.

2.1.5 Gruppe oder Team?

Häufig wird in der Literatur neben den schon genannten Begriffen auch das *Team* erwähnt: *„Ein Team ist eine Arbeitsgruppe, deren Mitglieder den Willen haben, ein gemeinsames Ziel zu erreichen"*(Teufel et al., 1995, S.10). Eine ähnliche Definition ist in Zimbardo & Gerrig (2000, S.723) zu finden: *„Eine Gruppe wird zum* Team*, wenn die Beiträge von zwei oder mehr Individuen zugunsten des erfolgreichen Erreichens eines gemeinsamen Zieles oder eines Auftrags koordiniert werden"*.

So ist ein Team auch eine Gruppe. Damit aus einer Gruppe aber ein Team wird, müssen weitere Eigenschaften erfüllt sein. So finden wir in der Literatur zwar verschiedene Aufzählungen von Teammerkmalen (wie bspw. gegenseitige Wertschätzung, gemeinsame und von allen Mitgliedern geteilte Ziele, Gleichberechtigung etc.), aber letztlich tragen auch all diese Aufzählungen nicht zu einer einheitlichen und deutlichen Abgrenzung zwischen den Begriffen bei (vgl. Kriz & Nöbauer, 2002, S.21ff; Kauffeld, 2001, Kap.2.2).

Nachfolgend finden daher die Begriffe *Gruppe*, *Arbeitsgruppe* (bzw. *Lerngruppe*) und *Team* weitgehend synonyme Verwendung und werden nur dort voneinander unterschieden, wo eine solche Unterscheidung notwendig ist. Letztlich bleibt es einer Gruppe selbst überlassen, ob sie sich als *Gruppe* oder als *Team* sieht, für die auf sie anzuwendenden Verfahren der Teamanalyse ist die Unterscheidung nicht relevant. Anders formuliert: *„Ein Team [ist] dann ein Team, wenn es sich selbst als Team definiert"* (Kauffeld, 2001, S.13). Betrachtungen zu Aspekten der Teamleitung (vgl. Mann, 2001, S.61f; Konradt & Hertel, 2002) sind in der vorliegenden Arbeit nicht relevant, da hier ausschließlich Gruppen beziehungsweise Teams mit einer flachen Hierarchiestruktur ohne explizite Führungsperson betrachtet werden.

2.2 Gruppenprozesse

Interaktion als integrales Merkmal sämtlicher Gruppen impliziert kommunikative Handlungen zwischen den einzelnen Gruppenmitgliedern, die auch als Gruppenpro-

zesse bezeichnet werden. Unter dem speziellen Aspekt der Zielerreichung, welcher ein wesentliches Merkmal von Arbeitsgruppen darstellt, sind die Prozesse *Kommunikation*, *Koordination* und *Kooperation* unabdingbar. Diese stellen grundlegende Elemente einer erfolgreichen Gruppenarbeit dar (vgl. TEUFEL ET AL., 1995, S.11).

2.2.1 Kommunikation

„Kommunikation ist die Verständigung mehrerer Personen untereinander" (TEUFEL ET AL., 1995, S.12). Kommunikation als Basis jeglicher Interaktion ist auch wesentlicher Bestandteil nahezu aller Gruppendefinitionen (s. 2.1) und somit ein wichtiges Merkmal, anhand dessen Gruppen von sozialen Aggregaten unterschieden werden können.

Diese Definition stellt sicher eine der einfachsten Festlegungen des Begriffs *Kommunikation* dar, sie beinhaltet aber das Wesentliche dessen, was im Kontext der vorliegenden Arbeit unter Kommunikation verstanden wird: Den Austausch von Informationen zwischen Kommunikationspartnern beziehungsweise Gruppenmitgliedern (vgl. (BROY & SPANIOL, 1999, S.373); (DORSCH, 1982, S.343)). Kommunikationsinhalte werden werden dabei stets über einen oder mehrere Kommunikationskanäle transportiert. Die an der Kommunikation beteiligten Kommunikationspartner kennzeichnen die Endpunkte eines solchen Kanals und werden in *Sender* (auch *Absender*; Quelle oder Ursprung der Kommunikation) und *Empfänger* (Ziel der Kommunikation) differenziert. Diese Definition von Kommunikation mag an dieser Stelle genügen, da sie alles Wichtige für das Verständnis der vorliegenden Arbeit umfasst. Bei Interesse sei auf einschlägige Literatur zu diesem Themenkomplex verwiesen.

2.2.2 Koordination

Da die Bildung von Arbeitsgruppen (und auch Lerngruppen) üblicherweise zu dem Zweck erfolgt, solche Ziele gemeinsam zu erreichen, die nur durch gemeinsame Anstrengungen aller erreicht werden können, muss auch die dazu notwendige Kommunikation so erfolgen, dass *„Reibungsverluste, die sich z. B. in Form von Doppelarbeit, erhöhtem Suchaufwand, Wartezeiten, Missverständnissen usw. ausdrücken, vermieden werden"* (HASENKAMP ET AL., 1994, S.20).

Diese spezielle Form der Kommunikation, die innerhalb der Gruppenarbeit zur Abstimmung aufgabenbezogener Tätigkeiten stattfindet, nennt man auch Koordination: *„Koordination bezeichnet jene Kommunikation, welche zur Abstimmung aufgabenbezogener Tätigkeiten, die im Rahmen von Gruppenarbeit ausgeführt werden, notwendig ist"*(TEUFEL ET AL., 1995, S.12).

Die Notwendigkeit zur Abstimmung der Aktivitäten der Gruppenmitglieder resultiert aus z. T. gegenseitigen Abhängigkeiten bei der Bearbeitung von Teilaufgaben, die sich beispielsweise aus der Nutzung gemeinsamer Ressourcen ergeben. Generell lässt sich festhalten, dass mit zunehmender Aufgabeninterdependenz auch die

Notwendigkeit von Koordination zunimmt, welche wiederum in einem erhöhter Kommunikationsbedarf resultiert.

2.2.3 Kooperation

Da Arbeitsgruppen zur Verfolgung gemeinsamer (Arbeits-)Ziele gebildet werden (oder sich in Selbstorganisation ihrer Mitglieder bilden), müssen die Tätigkeiten der einzelnen Mitglieder nicht nur koordiniert werden, sondern auch die zu erreichenden Ziele kommunikativ vereinbart werden. Diese spezielle Form der Kommunikation wird Kooperation genannt: *„Kooperation bezeichnet jene Kommunikation, die zur Koordination und zur Vereinbarung gemeinsamer Ziele notwenig ist"*(Teufel et al., 1995, S.12).

Die Abbildung 2.2 veranschaulicht die hierarchische Anordnung der soeben definierte Begriffe und verdeutlicht die zentrale Bedeutung der Kommunikation für alle weiteren innerhalb der Gruppe zur (erfolgreichen) Zielerreichnung notwendigen Aktivitäten.

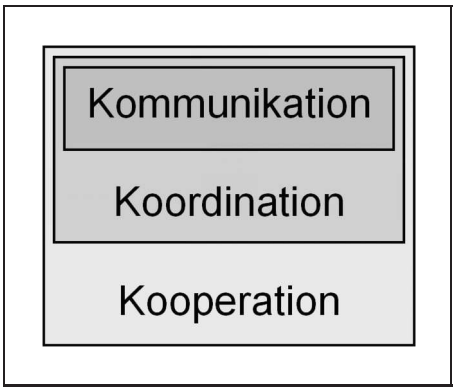

Abb. 2.2: Gruppenprozesse
(nach Teufel et al., 1995, S.11)

Funktionierende Kommunikation, Koordination und Kooperation zwischen allen Beteiligten sind für ein Funktionieren aller weiteren, nachgelagerten Gruppenprozesse unabdingbar, denn *„die Schwierigkeiten bei der Gruppenarbeit sind oft darauf zurückzuführen, dass der Austausch wesentlicher Informationen nicht funktioniert"*(Sader, 2002, S.147).

2.2.4 Kollaboration

Zusammenarbeit wird in der Literatur auch als Kollaboration bezeichnet und teilweise synonym mit Kooperation verwendet. Im Kontext der vorliegenden Arbeit bedeutet Kollaboration soviel wie gemeinsames Arbeiten, wohingegen Kooperation als aufeinander abgestimmtes Arbeiten aufgefasst werden kann. Bei Kollaborationen bearbeiten Mitglieder einer Gruppe ein Problem oder eine Aufgabe über weite

Strecken gemeinsam (und in der Regel auch zeitgleich), wohingegen in Kooperationen Teilaufgaben identifiziert, auf die Teammitglieder verteilt und von diesen weitgehend autonom bearbeitet werden, um letztlich die Teilergebnisse zu einem Gesamtergebnis zusammenzutragen.

Infolgedessen kann eine Kollaboration im Vergleich mit einer Kooperation durchaus als die intensivere Form der Zusammenarbeit bezeichnet werden, bedingt durch größere Phasen gemeinsamer Zusammenarbeit. Insbesondere für Lernprozesse ist kollaboratives Lernen relevant (s. Kap.2.2.6 und 5.4.2). Jedoch herrscht selbst in der Fachwelt nicht unbedingt Einigkeit darüber, ob das Lernen in der Gruppe nun kooperativ oder kollaborativ, kollektiv oder gar kompetativ erfolgt (vgl. HAAKE ET AL., 2004, S.1). In der vorliegenden Arbeit werden daher kooperatives Lernen und kollaboratives Lernen synonym betrachtet und lediglich dort voneinander unterschieden, wo eine Differenzierung notwendig und sinnvoll ist.

2.2.5 Gruppenarbeit

Kooperation, wie sie in Arbeitsgruppen stattfindet, wird als Gruppenarbeit bezeichnet und beinhaltet nach TEUFEL ET AL. *„die Summe aller aufgabenbezogenen Tätigkeiten, welche von Gruppenmitgliedern ausgeführt werden, um zielbezogene Aufgaben zu erfüllen und somit Gruppenziele zu erreichen"*(vgl. TEUFEL ET AL., 1995, S.11).

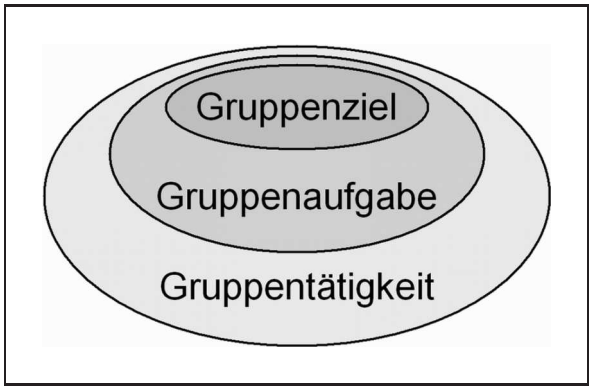

Abb. 2.3: Elemente der Gruppenarbeit
(nach TEUFEL ET AL., 1995, S.11)

Die aufgabenbezogene und zielorientierte Ausführung der Gruppentätigkeiten bedarf dabei der bereits erwähnten Gruppenprozesse. Damit aber überhaupt Gruppenarbeit stattfinden kann, müssen – neben der Möglichkeit der Kommunikation – weitere wichtige Voraussetzungen erfüllt sein. KRIZ & NÖBAUER haben folgende Bedingungen zusammengetragen (vgl. KRIZ & NÖBAUER, 2002, S.35ff):

- Entscheidungsdelegation beinhaltet Kriterien, unter denen eine Übertragung einer Problemlösung an eine Gruppe sinnvoll erscheint. Dies ist insbesondere dann erfüllt, wenn eine Aufgabe komplex ist und die Kompetenzen mehrerer

(z. T. unterschiedlich qualifizierter) Personen erfordert oder aber wenn eine gemeinsame Akzeptanz bzgl. einer Entscheidung erwünschenswert erscheint.

- Die Aufgabe muss so strukturiert sein, dass Gruppenarbeit überhaupt sinnvoll durchführbar ist. So wird unter anderem gefordert, dass die Aufgabe in Teilaufgaben zerlegbar ist und von mehreren Personen bearbeitet werden kann; zwischen den Teilaufgaben bestehen Interdependenzen, die Interaktionen erfordern. Ferner werden für die Problemlösung verschiedene Fähigkeiten und Qualifikationen benötigt. Auch die Attraktivität der Aufgabe wird als wichtiges Kriterium genannt, wobei wesentlich ist, dass die Bearbeitung im Team von den Beteiligten als in irgendeiner Form positiv empfunden wird.

- Technische und organisatorische Voraussetzungen bilden weitere Rahmenbedingungen, unter denen Aufgaben überhaupt erst an Gruppen übertragen werden können. Dazu werden neben rein ausführenden Tätigkeiten auch Koordination und Kooperation gefordert. Des weiteren sind Aufgabenvollständigkeit, Tätigkeitsspielraum, Anforderungsvielfalt und vor allem die Qualifikationen der Mitglieder relevant. Letztere müssen so qualifiziert sein, dass sie zur Problemlösung beitragen können.

Gruppenarbeit wird nur dann als sinnvoll erachtet, wenn die o.g. Voraussetzungen zu einem Großteil erfüllt sind (vgl. Kriz & Nöbauer, 2002, S.37).

2.2.6 Gruppenlernen

Neben der Gruppenarbeit, die sich längst als eine von mehreren Arbeitsformen in Organisationen und Unternehmen etabliert hat, gewinnt auch das Lernen in der Gruppe – oder auch *kooperatives/kollaboratives Lernen*[2] – immer mehr Bedeutung für den Wissenserwerb des Einzelnen, denn *„innerhalb von und zwischen Gruppen wird nicht nur neues Wissen generiert, sondern auch vorhandenes Wissen vernetzt und zur Verfügung gestellt"* (Kriz & Nöbauer, 2002, S.20). Gemeinsames Lernen bringt also dem Einzelnen wie der Gruppe Vorteile: *„Collaborative learning benefits a student by her observation of the activities of another student. It also benefits a group where its activities exceed activities performed by a set of individuals working alone"* (McManus, 1997, S.7).

Obwohl sich die vorliegende Arbeit auf die Computer-Unterstützung von Lerngruppen und Lernprozessen konzentriert, soll an dieser Stelle darauf hingewiesen werden, dass Lernen in der Gruppe ursprünglich ohne den Einsatz von Technologien stattfand: *„In der Pädagogik ist seit langem bekannt – ohne dass der Einsatz von Computern dabei thematisiert worden wäre – dass kooperative Lernformen dem individuellen Lernen oft überlegen sind. So führt passives Hören eines Vortrags oder isoliertes Lesen eines Buches bei den meisten (nicht bei allen!) Lernenden zu geringerem Lernerfolg als die aktive Erarbeitung eines Wissensgebietes durch die Diskussion in einer Gruppe, durch gemeinsames Erarbeiten von Aufgaben und mit*

[2] s. Kap.2.2.4

unterstützenden Hilfestellungen durch einen Tutor" (Wessner, 2001, S.196). Genaugenommen geht der Grundgedanke gemeinschaftlichen Lernens auf die pädagogische Reformbewegung zurück, welche sich gegen Ende des 19. Jahrhunderts formierte und nachfolgende didaktische Theorien wie den Konstruktivismus maßgeblich beeinflusst hat (s. Kap.5.4.1).

Mehr noch als bei der Gruppenarbeit wirkt sich Kommunikation in Form aktiver Teilnahme jedes einzelnen Mitglieds positiv auf den jeweiligen Lernprozess aus. So schreibt McManus weiter: *„...a student learns greatly when interacting with a more skilled student or teacher, especially through a discussion of their activities"* (McManus, 1997, S.7). Auch andere Autoren betonen die Vorteile des Lernens durch Diskurs (vgl. Konrad & Traub, 2005, S.180; Baker et al., 2001, S.89; Singley et al., 2000, S.146; Soller & Busetta, 2003, S.6).

Dennoch, man muss sich der Erkenntnis bewusst sein, dass die Vorteile kollaborativen Lernens nur dann zum Tragen kommen, wenn bestimmte Voraussetzungen durch die Lerngruppe und ihre Mitglieder erfüllt werden wie positive Wechselwirkungen zwischen den Gruppenmitgliedern, eine individuelle Verantwortlichkeit aller Teilnehmer für ihre jeweiligen Teilaufgaben, Informationsaustausch und Rückmeldungen (Feedback) untereinaner, ein kooperatives Miteinander (speziell in Konfliktsituationen) und eine permanente Reflexion der Gruppenprozesse durch die Gruppe selbst (s. Kap.2.1.4; Konrad & Traub, 2005, S.6; Grune & de Witt, 2004, S.27; Wessner, 2004, S.204f).

Zusammenfassend darf also zunächst festgestellt werden, dass Lernen in der Gruppe gegenüber dem individuellen Lernen gewisse Vorteile besitzen kann, die eine Unterstützung dieser Lernform rechtfertigen. Weitere Aspekte werden in Kapitel 5.4.2 betrachtet werden.

2.3 Gruppenstrukturen

In jeder Gruppe stehen die Mitglieder zueinander in Verbindung und interagieren miteinander.

Diese Verbindungen und Interaktionen unterliegen einem Entwicklungsprozess, der mit der Gruppenbildung beginnt, und werden als Gruppenstruktur bezeichnet, sobald sich eine gewisse Stabilität eingestellt hat: *„Das Muster funktionaler Beziehungen der Gruppenmitglieder, die unterschiedliche Positionen einnehmen, bildet die Gruppenstruktur"*(Zimbardo & Gerrig, 2000, S.723). Dabei lassen sich in der Regel mehr oder weniger stark ausgeprägte Unterschiede innerhalb einer Gruppe feststellen. Die Dimensionen, nach denen Gruppen strukturiert sein können, sind im Prinzip vielfältig, jedoch stellen soziometrische, Macht-, Kommunikations- und Rollenstrukturen die wohl wichtigsten Gruppenstrukturen dar (vgl. Mann, 2001, S.53). Für die vorliegende Arbeit sind insbesondere die Rollenstrukturen bedeutsam und werden im Anschluss an einen kurzen Überblick näher erläutert (s. Kap.2.5).

2.3.1 Soziometrische Strukturen

Soziometrische Strukturen oder auch Freundschaftsstrukturen beschreiben das aus Freundschaft, Gleichgültigkeit und Ablehnung bestehende Beziehungsgeflecht innerhalb von Gruppen. Insbesondere die Freundschaft zwischen Gruppenmitgliedern stellt einen wichtigen Faktor für die Gruppenproduktivität dar, da sie zum einen die Informationsverbreitung innerhalb der Gruppe positiv beeinflusst und überdies Produktivität und Effizienz der Gruppe erhöhen kann. Im Gegenzug aber kann sich zuviel Freundschaft auch kontraproduktiv auswirken, wenn sie zum dominierenden Faktor innerhalb einer Gruppe wird (vgl. Mann, 2001, S.53f).

Soziometrische Strukturen werden üblicherweise durch Beobachtung von Gruppen und Befragung der Gruppenmitglieder ermittelt und in sog. *Soziogrammen* dargestellt.

2.3.2 Machtstrukturen

Machtverhältnisse innerhalb einer Gruppe stellen eine weitere gängige Dimension dar, anhand derer eine Gruppe strukturiert werden kann. Macht gilt als Statussymbol und ist in der Regel mit diversen Vorteilen verbunden. Dorsch definiert Macht als *„die Chance, innerhalb einer sozialen Beziehung den eigenen Willen auch gegen Widerstreben durchzusetzen, gleichviel worauf diese Chance beruht"* (Dorsch, 1982, S.398).

In der Literatur werden fünf Grundformen der Macht genannt: legitime Macht, Belohnungsmacht, Zwangsmacht, Vorbildmacht und Expertenmacht (vgl. Eunson, 1990, S.407; Sader, 2002, S.67; Dorsch, 1982, S.398). Neben einem höheren Ansehen sind Gruppenmitglieder mit hoher Machtstellung auch stärker an der Gruppenaufgabe beteiligt und werden üblicherweise auch soziometrisch bevorzugt. In der Folge steigt auch die eigene Zufriedenheit – nicht zuletzt auch durch die Möglichkeit, die Macht einsetzen zu können, indem beispielsweise Entscheidungsprozesse der Gruppe beeinflusst und zum eigenen Vorteil entschieden werden können.

2.3.3 Kommunikationsstrukturen

Die Kommunikationskanäle zwischen Gruppenmitgliedern werden in ihrer Gesamtheit als Kommunikationsstruktur der Gruppe bezeichnet. Diese Struktur gibt Auskunft darüber, welche Gruppenmitglieder miteinander kommunizieren, und beschreibt somit, wie innerhalb einer Gruppe Informationen ausgetauscht werden und wie – im Rahmen der Gruppenarbeit – die weiteren Gruppenprozesse (Koordination und Kooperation) durchgeführt werden.

Die Kommunikationsstruktur ist nicht nur grundlegend für die soziometrische Struktur und die Machtstruktur einer Gruppe, sondern beeinflusst auch maßgeblich die Motivation und Produktivität der Gruppe und ihrer Mitglieder (vgl. Haake et al., 2004, S.43f). Bei Untersuchungen von Kleingruppen hat Leavitt erstmals

diese Auswirkungen untersucht (vgl. HARE ET AL., 1955, S.400ff; HARE, 1962, S.272ff; MANN, 2001, S.56ff; HAAKE ET AL., 2004, S.43f). Verschiedene der von ihm eingesetzten Kommunikationsstrukturen sind in Abbildung 2.4 dargestellt.

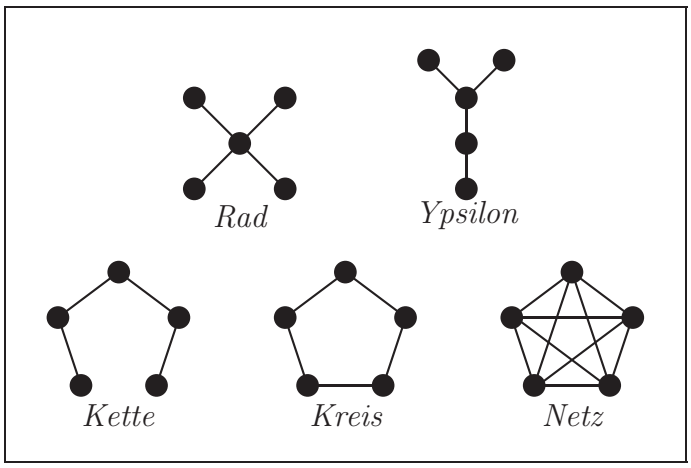

Abb. 2.4: Kommunikationsnetze in Kleingruppen zu je fünf Personen (nach LEAVITT, 1951)

Im Rahmen seiner Untersuchungen entdeckte LEAVITT beispielsweise, dass der Zentralisierungsgrad einer Kommunikationsstruktur von wesentlicher Bedeutung für die Effizienz der Gruppe ist: *„die Kreis- und Kettengruppe [ist] weniger effizient als die Rad- und Y-Gruppe [...], da sie mehr Botschaften benötigen, um zu Lösungen zu kommen"* (MANN, 2001, S.57). Insofern ist eine Kommunikationsstruktur zu unterstützen, die alle Teammitglieder gleichermaßen berücksichtigt, ohne eines davon zu bevorzugen. Dieser Aspekt wurde bei der Realisierung des Vitaminl-Systems entsprechend berücksichtigt: Sämtliche Gruppenkommunikation wird stets allen Mitgliedern zugänglich gemacht (s. Kap.8.1.1.2).

2.3.4 Rollenstrukturen

Eine weitere wichtige Dimension stellt die Aufgabenverteilung innerhalb einer (Arbeits-)Gruppe dar und damit verbunden die Erwartungshaltungen der Gruppenmitglieder an denjenigen, der zur Erfüllung einer bestimmten Aufgabe beiträgt. Diese Erwartungshaltung wird auch als *Rolle* bezeichnet: *„Eine Rolle ist ein sozial definiertes Verhaltensmuster, das von einer Person, die eine bestimmte Funktion in einer Gruppe hat, erwartet wird"*(ZIMBARDO & GERRIG, 2000, S.723). Die Person, die eine solche Rolle wahrnimmt, wird auch *Rollenträger* genannt.

Eine Rollenstruktur beschreibt also die Verteilung von gewissen Rollen innerhalb einer Gruppe: *„Die Rollen- oder Arbeitsstruktur ist das Aufgaben- oder Verantwortlichkeitsmuster der Mitglieder innerhalb der Gruppe; sie stellt die Arbeitsteilung oder Rollenverteilung der Gruppe dar"* (MANN, 2001, S.59). In formalen Arbeitsgruppen mit hohem Organisierungsgrad entspricht dies der Aufgabenspezialisierung der einzelnen Mitglieder, in informellen, eher unstrukturierten Gruppen

bezieht sich die Rollenstruktur auf die zur Zielerreichung notwendige Rollendifferenzierung. Dabei wird der Begriff der Rolle durchaus unterschiedlich interpretiert und kann – je nach Sichtweise – einen Persönlichkeitstyp mit bestimmten Verhaltenseigenschaften bezeichnen oder auch eine im Rahmen von Gruppenarbeit zu erfüllende Arbeitsfunktion benennen. Alle für eine bestimmte Sichtweise relevanten Rollen werden in einem sog. *Rollenmodell* zusammengefasst.

Aufgrund der Vielfältigkeit der Sichtweisen und der Relevanz für die vorliegende Arbeit werden Rollen und Rollenmodelle im Kapitel 2.5 ausführlicher betrachtet. Zunächst aber folgt eine Übersicht über gängige Verfahren, mit deren Hilfe versucht wird, Strukturen in Teams zu bestimmen.

2.4 Teamdiagnose

Die Teamdiagnose als Begriff aus der Psychologie soll mit Hilfe ihrer Instrumente Sachverhalte innerhalb einer Gruppe offenlegen, die beim einfachen Betrachten einer Gruppe oder einer Gruppendiskussion nicht mit bloßem Auge erkennbar sind: *„Die Diagnose ist der Ausgangspunkt jedes Teamentwicklungsprozesses, der auf eine Problemlösung abzielt"* (Kauffeld, 2001, S.50).

2.4.1 Ziele

Teamdiagnose dient zunächst der Gewinnung von Information über eine Gruppe, ihre Strukturen und ihre Prozesse. Sie erfüllt aber neben der reinen Bestandsaufnahme auch Zwecke der Bedarfsermittlung von Teamentwicklungsmaßnahmen und bietet somit einen Ausgangspunkt für Prozesse der Teamentwicklung. Vor allem kann eine Teamdiagnose angewendet werden, um Schwachstellen in einem Team aufzudecken und Ansatzpunkte für Verbesserungen aufzuzeigen. Die Durchführung einer Teamdiagnose nach erfolgten Teamentwicklungsmaßnahmen kann der Überprüfung beziehungsweise Evaluierung der durchgeführten Maßnahmen dienen.

Im Kontext dieser Arbeit stellt die Teamdiagnose eine Möglichkeit dar, mit geeigneten Verfahren und Methodiken Information über die gegenwärtige Situation einer Gruppe bezüglich ihrer Rollenstruktur zu gewinnen. Dabei findet eine Beschränkung auf virtuelle Teams statt, die kurzzeitig und synchron innerhalb des Themenbereichs der objektorientierten Software-Entwicklung zusammenarbeiten.

2.4.2 Anforderungen

Teamdiagnose im Allgemeinen und die dabei verwendeten Instrumente im Speziellen müssen verschiedene Anforderungen erfüllen, damit ihr Einsatz den Ansprüchen wissenschaftlicher Forschung gerecht wird. Sowohl Kauffeld 2001 (S.50ff) als auch Zimbardo & Gerrig 2000 (S.15ff) nennen Kriterien, die Instrumente der

Sozialpsychologie im Allgemeinen und der Teamdiagnose im Speziellen erfüllen müssen:

- Theoretisch fundiertes Modell:
  Der Ausgangspunkt – und damit unverzichtbare Grundlage – jeglicher Teamdiagnose wird durch ein vollständiges und widerspruchsfreies Modell definiert, welches in der Lage ist, die für eine bestimmte Fragestellung oder Betrachtung relevanten Strukturen oder Prozesse innerhalb eines Teams aufzuzeigen. Zu solch einem Modell gehören unverzichtbarerweise auch die Einflussfaktoren, die innerhalb des Modells auf die Strukturen einwirken beziehungsweise Änderungen in den Strukturen bewirken. Das Modell bildet nicht nur den Rahmen der diagnostischen Tätigkeit und damit überhaupt die Möglichkeit, bestimmte Effekte in einer Gruppe zu erkennen, zu beschreiben und zu erklären, sondern darüber hinaus auch die Basis, um erkannte Effekte zielgerichtet zu beeinflussen.

- Integration relevanter Aspekte:
  Als Folgerung der im vorigen Punkt geforderten Modellvollständigkeit ergibt sich zwangsläufig die Möglichkeit, alle – für eine bestimmte Fragestellung – relevanten Aspekte mit den zur Verfügung stehenden Diagnoseinstrumenten erfassen zu können. Bei der Wahl der einzusetzenden Instrumente ist – neben der vollständigen Erfassung und der prinzipiellen Anwendbarkeit – auch der Detailiertheitsgrad zu beachten.

- Universelle Einsetzbarkeit:
  Die in der Literatur häufig gestellte Forderung nach einer Art Allgemeingültigkeit beziehungsweise universellen Anwendbarkeit eines teamdiagnostischen Instruments ist prinzipiell begrüßenswert, ermöglicht sie es doch, ein solches Verfahren in ganz unterschiedlichen Kontexten auf unterschiedliche Gruppen anzuwenden (vgl. Kauffeld, 2001, S.114). Trotzdem wird sich dieser Anspruch stets dem zugrundeliegenden Modell und dem Kontext, in welchem es Anwendung findet, unterordnen müssen, so dass das Kriterium der universellen Einsetzbarkeit korrekterweise eine Forderung nach universeller Einsetzbarkeit innerhalb des vereinbarten Modells darstellt.

- Ökonomische Einsetzbarkeit:
  Die Ökonomie von Diagnoseinstrumenten umfasst Aspekte sowohl der Anwendbarkeit als auch der Auswertbarkeit. Die ökonomische Anwendbarkeit eines Instruments bedeutet eine möglichst geringe Einflussnahme auf die zu betrachtende Situation, auf die zu analysierende Gruppe und damit auch auf die zu erfassenden Einflussfaktoren. Geringe Störungen der Gruppe und ihrer Mitglieder wirken sich nicht nur positiv auf die wiederholte Anwendbarkeit eines Diagnoseinstruments aus, sondern können auch maßgeblich für die Akzeptanz seitens der Teilnehmer sein. Die ökonomische Auswertbarkeit fordert Methodiken, mittels derer die Auswertung der erfassten Daten mit möglichst geringem Ressourceneinsatz erfolgen kann. Im Idealfall erfolgt der Einsatz eines teamdiagnostischen Instruments folglich ohne störende Eingriffe in die

zu beobachtende Situation (vgl. KAUFFELD, 2001, S.114). Die anschließende Auswertung der Resultate findet dabei weitestgehend automatisiert statt.

- Objektivität:
  „Die Grundforderung an die wissenschaftliche Forschung ist die Forderung nach Objektivität" (ZIMBARDO & GERRIG, 2000, S.16). Damit verbunden ist weithin die Nachprüfbarkeit durch andere Forscher (*intersubjektive Nachprüfbarkeit*). Die Objektivität eines Instruments kennzeichnet somit dessen Unabhängigkeit vom Anwender (vgl. KAUFFELD, 2001, S.115). Die Sicherstellung dieser Forderung kann gewährleistet werden durch eine eindeutige und präzise Definition der Variablen, durch die Verwendung standardisierter Erhebungsverfahren und durch den Einsatz vorbeugender Maßnahmen gegen Verfälschungen der gewonnenen Daten.

- Zuverlässigkeit:
  „Reliabilität bezeichnet die Genauigkeit der Messung. Sie ist dann in vollem Umfang gegeben, wenn bei Wiederholung der Erhebung unter denselben Bedingungen identische Resultate erzielt werden, wenn also die Messwiederholung zu konsistenten oder stabilen Resultaten führt" (ZIMBARDO & GERRIG, 2000, S.19). Unter der Annahme, dass es einen objektiven, *wahren* Wert gibt, beschreibt die Reliabilität den Grad der Übereinstimmung zwischem diesem wahren Wert und dem gemessenen Wert. Ein Reliabilitätswert von 1 bedeutet, dass Messwert und wahrer Wert identisch sind; das eingesetzte Messverfahren misst das zugehörige Kriterium in diesem Fall exakt.

- Validität:
  Die Validität beschreibt, ob und wie gut ein Instrument das erfasst, was es zu erfassen vorgibt: *„Validität bedeutet, dass das Verfahren tatsächlich das psychologische Merkmal oder die Variable misst, die es – der Erwartung nach – messen soll"* (ZIMBARDO & GERRIG, 2000, S.19). Als Gütekriterium beschreibt die Validität den Grad der Gültigkeit einer wissenschaftlichen Feststellung und damit auch deren Belastbarkeit.

- Akzeptanz:
  Die Beteiligungsquote wird in der Regel als Indiz für die Akzeptanz eines Diagnoseinstruments herangezogen und gibt Auskunft über den möglichen Erfolg oder Misserfolg einer durchgeführten Diagnose. Da diese von der Beteiligung möglichst vieler Mitglieder abhängt, empfiehlt es sich bereits im Vorfeld, eine möglichst große Akzeptanz unter den Teilnehmern mittels geeigneter Maßnahmen zu erreichen. Dies kann sowohl durch umfassende Information über das Verfahren und seine Konsequenzen erfolgen, aber auch durch Einbeziehung der Beteiligten in die Planung und Vorbereitung. Letztlich besteht bei geringer Akzeptanz die Gefahr mangelnder Teilnahme, was die Aussagekraft der Ergebnisse stark mindern kann.

- Anwendbarkeit:
  Die Anwendbarkeit eines Messinstruments fordert letztlich, dass es im täglichen Einsatz auch ohne begleitenden Experten verwendbar ist. Im Falle der

Teamdiagnose bedeutet dies, dass jedes Teammitglied in die Lage versetzt wird, das Instrument ohne Hilfe zu bedienen und auch die richtigen Schlüsse aus den Ergebnissen zu ziehen.

Es bleibt anzumerken, dass der Anforderungskatalog als Gesamtes eine Menge von Eigenschaften bildet, von denen einige wünschenswert, andere hingegen unverzichtbar sind. Insbesondere das Vorhandensein eines Modells sowie die Forderungen nach Objektivität und Validität stellen Kriterien dar, die zwingend im Kontext ernsthafter wissenschaftlicher Forschung erfüllt sein müssen, wohingegen beispielsweise auf die universelle Einsetzbarkeit unter Umständen verzichtet werden kann. So ist ein Verfahren, dass speziell in Kleingruppen Anwendung findet, nicht notwendigerweise auch auf Individuen oder Großgruppen anwendbar. Insgesamt sollte ein teamdiagnostisches Instrument den genannten Anforderungen weitestgehend genügen. Nur dann sind verwertbare Ergebnisse zu erwarten.

2.4.3 Instrumente

Die Instrumente der Teamdiagnose stellen die methodischen Zugänge zur Erfassung von Daten dar, die für die Diagnose relevant sind, unabhängig davon, ob die zugrundeliegenden Untersuchungen unter Laborbedingungen oder im Rahmen von Feldstudien durchgeführt werden. In der Teamdiagnose geht es vornehmlich um die Erfassung menschlichen Verhaltens und genau dort liegt auch die Herausforderung: Es muss Zugang zu den menschlichen Verhaltensweisen gefunden werden, das heißt interne Ereignisse und Prozesse müssen äußerlich erkennbar gemacht werden. Unter der Vielzahl von Instrumenten stellen psychologische Tests, Beobachtungen und Befragungen sicherlich die wichtigsten Zugänge dar.

2.4.3.1 Psychologische Tests

Ein psychologischer Test bezeichnet *„ein wissenschaftliches Routineverfahren zur Untersuchung eines oder mehrerer empirisch abgrenzbarer Persönlichkeitsmerkmale mit dem Ziel einer möglichst quantitativen Aussage über den relativen Grad der individuellen Merkmalsausprägung"* (Zimbardo & Gerrig, 2000, S.26). Es wird dabei versucht, mittels Fragen, Aufgaben oder Aktivitäten bestimmte Reaktionen bei einem Probanden hervorzurufen, die als Indikatoren für bestimmte psychologische Funktionen angenommen werden.

Die mit solchen Tests erzielten Resultate bedürfen in der Regel der Interpretation und setzen entsprechende Schulung und Erfahrung des Auswertenden voraus. Der verhältnismäßig hohe Aufwand zur Durchführung psychologischer Test kann in Ergebnissen mit hoher Genauigkeit und Detailiertheit resultieren, jedoch – und dies ist einer der Hauptkritikpunkte – sind solche durch Interpretation gewonnenen Resultate stets sehr subjektiv und vom Geschick und der Erfahrung des Diagnostikers abhängig.

2.4.3.2 Beobachtungen

Die Beobachtung im Sinne von Verhaltensbeobachtung im Rahmen wissenschaftlicher Forschung stellt eine der wichtigsten Möglichkeiten dar, etwas über andere Personen zu erfahren. Im Gegensatz zum alltäglichen Beobachten erfolgt das wissenschaftliche Beobachten jedoch weitaus geplanter, präziser und systematischer unter Anwendung von Beobachtungstechniken. Beobachtungen können direkt, also mit bloßem Auge durchgeführt werden, falls das relevante Verhalten klar erkennbar und zugänglich ist, sie können aber auch vermittelt unter Zuhilfenahme spezieller Ausrüstung oder Instrumente erfolgen, falls eine direkte Beobachtung nicht möglich ist oder eine präzisere Registrierung des beobachteten Verhaltens erwünscht ist. Falls während einer Beobachtung kein Versuch des Beobachters zur Veränderung der Situation beziehungsweise des Verhaltens erfolgt, spricht man auch von natürlicher oder naturalistischer Beobachtung (vgl. Zimbardo & Gerrig, 2000, S.24).

Die Qualität einer jeden Beobachtung ist in starkem Maße abhängig von der Sorgfalt des Beobachters. Er muss zunächst in der Lage sein, relevante von irrelevanten Daten zu unterscheiden. Oftmals müssen die erfassten Daten im Zuge der Erfassung auch bestimmten Kategorien zugeordnet werden. Die Beobachtung als Instrument der Datenerhebung setzt also einen gut geschulten Beobachter voraus.

Im Anschluss an eine Beobachtung findet in der Regel eine Transkription des beobachteten Verhaltens (also der erhobenen Daten) statt, wodurch die so gewonnenen Daten überhaupt erst erfassbar, messbar und auswertbar gemacht werden. Der Aufwand, der für solch eine Transkription notwendig ist, darf nicht vernachlässigt werden.

Das wissenschaftliche Beobachten bedient sich auch des Messens, also der Erfassung messbarer Daten: *„Das Messen kann als Zuordnung von Zahlen zu Objekten verstanden werden. Dabei sollen sich in den zugeordneten Zahlen die Relationen, die zwischen den Objekten bestehen, widerspiegeln"* (Dorsch, 1982, S.415). Die in der Psychologie verwendete Bedeutung geht also über die reine Erfassung von Zahlenwerten im Sinne eines physikalischen Messbegriffs hinaus (vgl. Zimbardo & Gerrig, 2000, S.22), umschließt aber auch diesen, insbesondere dann, wenn die Beobachtung mittels geeigneter (Mess-)Instrumente erfolgt.

2.4.3.3 Befragungen

Die Befragung als Instrument der Teamdiagnose existiert in vielen unterschiedlichen Varianten, denen gemeinsam ist, dass der Diagnostiker mit dem Teammitglied in einen Dialog eintritt, mit dem Ziel der Informationsgewinnung aus den Antworten und Äußerungen des Teammitglieds. Diese Information lässt sich durch bloße Beobachtung nicht oder nur sehr schwer gewinnen. So ist es beispielsweise nur durch Befragungen möglich, Information über Glauben, Einstellungen, Gefühle, Motive und Persönlichkeiten von Menschen zu gewinnen (s. Zimbardo & Gerrig, 2000, S.25). Eine Befragung kann in direkter Form in Interviews erfolgen, sie kann aber auch indirekt durch den Einsatz von Fragebögen in Abwesenheit des

Diagnostikers durchgeführt werden. Ferner kann sich eine Befragung auf ein einzelnes Teammitglied beziehen (individuelle Interviews) oder aber auch das gesamte Team beziehungsweise Teile davon betreffen (Gruppeninterviews, -diskussionen).

Die diversen Arten der Befragung unterscheiden sich ferner bezüglich des Aufwands, der betrieben werden muss, um die so erhobenen Daten auszuwerten. Während standardisierte Fragebögen im Extremfall automatisch (durch entsprechende Computerprogramme) ausgewertet und sogar interpretiert werden können, werden beispielsweise im Fall von Interviews mit offenen Fragen geschulte Experten benötigt, die die gewonnenen Daten mit zum Teil erheblichen Zeitaufwand auswerten und interpretieren müssen.

Auch hinsichtlich der Validität ist diese Methodik nicht ganz unkritisch: Versuchspersonen können durchaus falsche Antworten geben, weil sie nicht ihre wahren Gefühle preisgeben möchten, weil Erinnerungen an Vergangenes ungenau sind oder auch, weil die Fragen nicht richtig verstanden worden sind. So müssen weitere Anstrengungen unternommen werden, um die Validität einer Befragung sicherzustellen.

Neben den genannten Datenerhebungsverfahren ist es manchmal auch möglich, auf schon vorhandene Daten (wie bspw. betriebliche Dokumente in Form von Aktennotizen und Protokollen) oder betriebliche Vorgänge und Abläufe zurückzugreifen und diese ebenfalls zur Analyse heranzuziehen.

2.4.4 Teamdiagnostische Verfahren

Die Verwendung (und teilweise Kombination) der genannten Werkzeuge hat eine Vielzahl teamdiagnostischer Verfahren hervorgebracht. In der Literatur erfolgt eine grundsätzliche Differenzierung nach dem Aspekt der Teamdiagnose in prozess- und strukturanalytische Verfahren (vgl. Kauffeld, 2001, S.52ff).

2.4.4.1 Prozessanalytische Verfahren

Nach Dorsch ist die Prozessanalyse ein *„freier oder an Raster bzw. skalierenden Verfahren gebundener Versuch, die persönliche und sozialpsychologischen Aspekte eines Gruppenverlaufs (meist einer zeitlichen Einheit) zu beschreiben, etwa in Kategorien von Sympathie, Vertrauen, Machtausübung, Entwicklung von Kohäsion, Kooperationsfähigkeit etc."* (Dorsch, 1982, S.511). Im Rahmen der Kleingruppenforschung wurden einige standardisierte Verfahren zur Verhaltensbeobachtung beziehungsweise der Analyse von Prozessen entwickelt, mittels derer Verhalten in Kleingruppen klassifizierbar geworden ist.

2.4.4.1.1 Interaction Process Analysis Eines der ersten Verfahren ist die *Interaction Process Analysis* (IPA; vgl. Bales, 1950): *„Der Verlauf und die Dynamik des Gruppenprozesses werden beschrieben, indem die Funktionen der geäußerten Inhalte einer Gruppendiskussion nach bestimmten (Verhaltens-)Kategorien ein-*

geordnet werden" (KAUFFELD, 2001, S.54). Die Kategorien des verwendeten Bewertungssystems, *Bales' set of interaction categories*, sind der nachfolgenden Abbildung 2.5 entnehmbar (vgl. BALES, 1950, S.8ff; HARE, 1962, S.66ff).

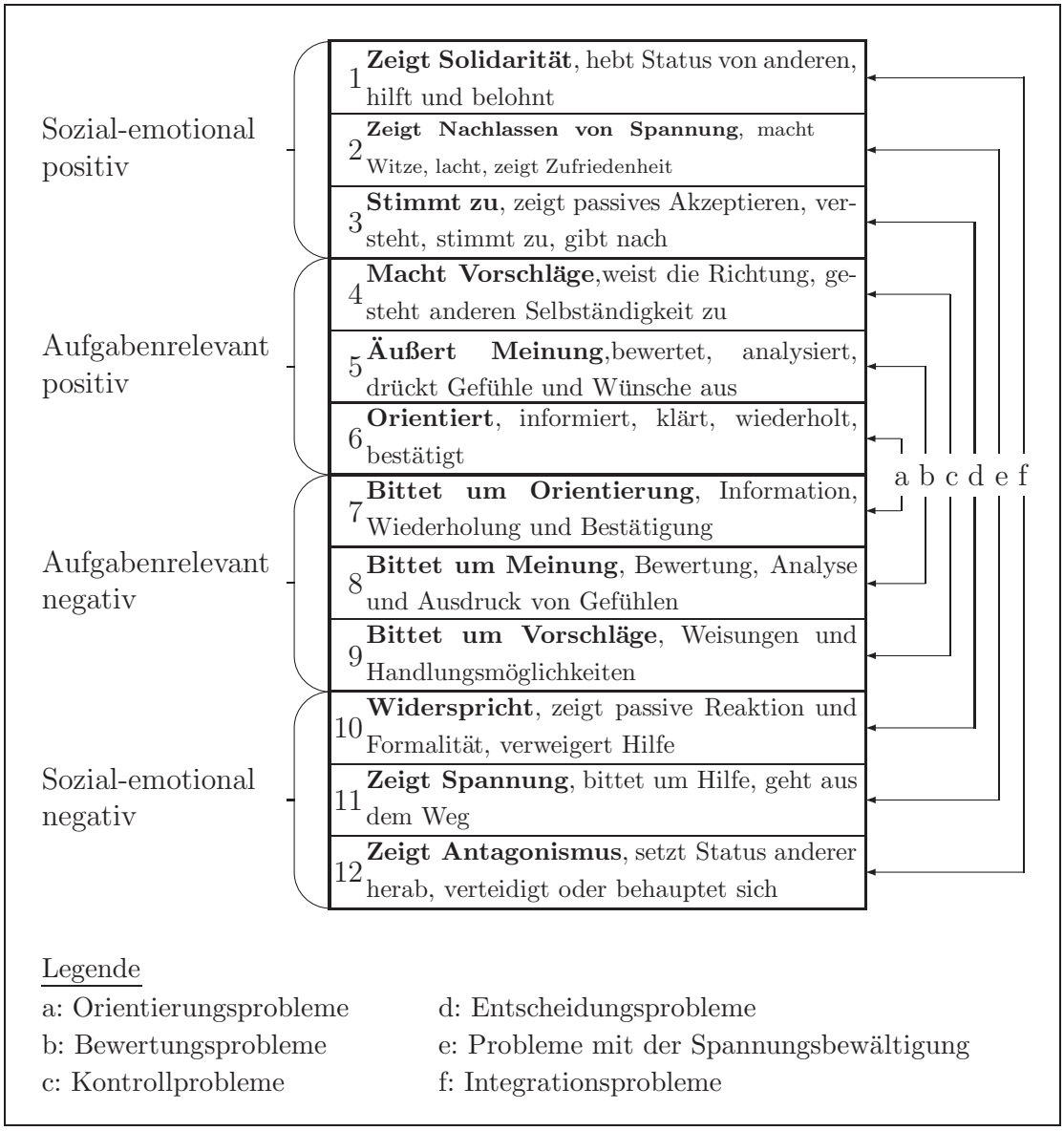

Abb. 2.5: Kategorien sozialer Interaktion
(nach BALES, 1950, S.9)

Bales nahm an, der Erfolg einer Gruppe hänge von zwei grundlegenden Faktoren ab: Wie gut löst eine Gruppe die anstehenden Aufgaben und wie gut gelingt es ihr, eine zufriedenstellende Gruppenatmosphäre zu schaffen. Dazu führte er Versuche mit Kleingruppen durch, die sich in einem speziellen Beobachtungszimmer trafen und zu einem gegebenen fiktiven Problem eine Diskussion mitsamt anschließender Entscheidungsfindung durchzuführen hatten. Dabei wurden die Gruppenmitglieder wissentlich durch einen Einwegspiegel beobachtet: Sämtliche Äußerungen der Gruppenmitglieder wurden aufgezeichnet und anhand der in Abbildung 2.5 ge-

nannten Kategorien notiert. Anschließend wurden die Ergebnisse auf Lochkarten übertragen und maschinell ausgewertet: *„Extensive processing of the data is facilitated by transferring all scores on the tapes to I.B.M. punch cards, one card per score"* (Bales, 1950, S.13). Mittels statistischer Verfahren wurden die Ergebnisse aufbereitet, ausgewertet und unter dem Aspekt der jeweiligen Fragestellung interpretiert. Mittels dieser Vorgehensweise konnten beispielsweise verschiedene Teamrollen identifiziert werden (s. Kap.2.5.1).

Dieses Verfahren ist nicht nur deshalb interessant und erwähnenswert, weil es die Grundlage weiterer teamdiagnostischer Verfahren darstellt, sondern vor allem, weil die Idee, Gruppen und ihre Teilnehmer anhand einer Taxonomie sozialer Interaktionen (vornehmlich Kommunikation) zu analysieren und somit beispielsweise auf Rollen zu schließen, in der vorliegenden Arbeit aufgegriffen und in den Kontext der objektorientierten Programmierung transferiert wird.

2.4.4.1.2 System for the Multiple Level Observation of Groups Auf der Grundlage von IPA entwickelte Bales & Cohen später ein weitaus umfassenderes Analysesystem, SYMLOG (*SYstem for the Multiple Level Observation of Groups*), das neben der Rollendimension (Aufgabenbezogenheit vs. Sozioemotionalität) noch zwei weitere Dimensionen, die Statusdimension (Einfluss vs. Macht) und die Akzeptanzdimension (Freundlichkeit vs. Unfreundlichkeit), beinhaltet (vgl. Bales & Cohen, 1982; Kauffeld, 2001, S.54f). Die Weiterentwicklung besteht in der Berücksichtigung verschiedener Kommunikationskanäle, die auch nicht-verbale Kommunikation (in Form von Mimik und Gestik) mit einbezieht. Die Mehrdimensionalität beinhaltet aber nicht ausschließlich auf die Möglichkeit, verschiedene Stufen des Verhaltens zu erfassen, sondern umschließt auch verschiedene Stufen der geäußerten Inhalte: In der Kodierung wird berücksichtigt, ob sich eine Äußerung beispielsweise auf die Gruppe, ein anderes Gruppenmitglied oder eine äußere Situation bezieht.

2.4.4.1.3 Weitere Verfahren der Interaktionsanalyse Weitere gängige Verfahren sind die ebenfalls auf IPA basierende Konferenzkodierung (KONFKOD von Fisch, 1994; vgl. Kauffeld, 2001, S.55), das Kasseler-Kompetenz-Raster (KKR) zur Ermittlung der Kompetenz von Mitarbeitern im Rahmen von Gruppendiskussion (Kauffeld, 2001, S.55) oder auch die kognitiven Karten (*cognitive maps*) von Axelrod (1976) zur Darstellung der inhaltlichen Ebene eines Interaktionsprozesses.

Zwar liefern die prozessanalytischen Verfahren erfahrungsgemäß sehr detailierte Ergebnisse, unter ökonomischen Aspekten sind diese Verfahren jedoch kritisch zu betrachten, da sie – bedingt durch Codierungen und Transkriptionen der Aufzeichnungen – mit einem vergleichsweise hohen Aufwand verbunden sind. Daher kommen in der Praxis bevorzugt strukturanalytische Verfahren zum Einsatz.

2.4.4.2 Strukturanalytische Verfahren

Eine Strukturanalyse hat die Erfassung und Abbildung der in einer Gruppe existierenden Strukturen zum Ziel. Dazu werden üblicherweise Fragebögen, Rating-Bogen und Adjektivlisten verwendet, die weitgehend standardisiert sind. Diese Instrumente sind sowohl leicht zu handhaben als auch zeit- und ressourcenökonomisch anwendbar und bedürfen nur geringer Vorbereitung und Training durch den Anwender beziehungsweise Beobachter. Darin liegen auch die wesentlichen Vorteile gegenüber prozessanalytischen Verfahren begründet.

Das Ergebnis einer Strukturanalyse kann also als eine Art Schnappschuss einer Gruppe zu einem bestimmten Zeitpunkt verstanden werden, der jedoch nur ein grobes, wenig detailgetreues Bild der Gruppe enthält. Trotzdem stellt die Strukturanalyse eine effiziente Methode für die Langzeituntersuchung von Gruppenstrukturen und -prozessen dar. Dabei ist anzumerken, dass die Erfassung (und Abbildung) von Prozessen beziehungsweise Änderungen innerhalb von Gruppenstrukturen nur durch Wiederholungsmessungen möglich ist.

2.4.4.2.1 Klassifikationsraster Klassifikationsraster stellen den Versuch dar, verschiedene Konzepte der Gruppenarbeit anhand sozialer Benchmarks vergleichbar zu machen. Als vornehmliches Kriterium zur Typisierung von Gruppenarbeitsformen dient oftmals der Autonomiegrad einer Gruppe. Die Erhebung erfolgt in der Regel anhand eines Fragebogens durch den Gruppensprecher oder ein ausgewähltes Mitglied der Gruppe. Obwohl die Kategorien oftmals eher abstrakt gehalten sind und eine Operationalisierung der Merkmale vermisst wird, kann das Ergebnis durchaus verwendet werden, um innerhalb einer Gruppe eine Diskussion über die Gruppenarbeit anzuregen und Vergleiche anhand konkreter Kriterien durchzuführen (vgl. Kauffeld, 2001, S.60f).

2.4.4.2.2 Organisationsdiagnostische Verfahren Mitarbeiterbefragungen auf der Basis von Skalen zur "Zusammenarbeit mit Kollegen“ existieren in einer Vielzahl von Varianten. Anhand weniger, sehr allgemein gehaltener Fragen beziehungsweise zu bewertender Aussagen wie *„Man hält in der Abteilung zusammen."* kann zwar ein genereller Bedarf an Teamentwicklungsmaßnahmen festgestellt werden, jedoch sind die Ergebnisse – wie auch die Fragen – in der Regel sehr allgemein und liefern keine Anhaltspunkte für Teamentwicklungsmaßnahmen (vgl. Kauffeld, 2001, S.61f).

2.4.4.2.3 Soziometrie Das Hauptziel soziometrischer Verfahren liegt in der Analyse zwischenmenschlicher Beziehungen, üblicherweise in der Form von Sympathie-/Antipathiestrukturen (s. Kap.2.3.1). Soziometrische Verfahren liefern Außenstehenden einen schnellen Überblick über eine Gruppe und ihre soziometrischen Strukturen und beantworten Fragestellungen wie beispielsweise *„Wer hat mit wem am meisten Kontakt?"*, *„Gibt es Außenseiter in der Gruppe?"* oder auch *„Haben sich Untergruppen in der Gruppe gebildet?"*. So sinnvoll die Ergebnisse

für Außenstehende sein können, so wenig Informationen ergeben sich in der Regel für die Gruppenmitglieder. Sie erhalten üblicherweise keine neuen Erkenntnisse, vielmehr besteht die Gefahr, dass aus der Offenlegung der Ergebnisse Stigmatisierungseffekte folgen, durch die Teamentwicklungsprozesse eher behindert als unterstützt werden (vgl. KAUFFELD, 2001, S.62f).

2.4.4.2.4 Adjektiv-Ratingbogen Mit Adjektiv-Ratingbögen erhalten Gruppenmitglieder die Möglichkeit, sich selbst und die anderen Mitglieder anhand von Adjektivlisten zu beurteilen. So wird beispielsweise im SYMLOG-Bogen (in Anlehnung an das gleichnamige prozessanalytische Verfahren; s. Kap.2.4.4.1.2) auf einer Skala notiert, wieweit ein gegebenes Adjektiv (z.B. kooperativ, interessiert) auf eine bestimmte Person zutrifft. Die Auswertung und Interpretation eines SYMLOG-Bogens soll Erkenntnisse bzgl. der Dimensionen Rolle, Status und Sympathie liefern. In der Subjektivität sowohl der Interpretation der Ergebnisse durch den Diagnostiker als auch des Zustandekommens der Ergebnisse durch den Teilnehmer liegt die wesentliche Kritik an diesen Verfahren: Sowohl das Zustandekommen des Urteils als auch das zugrundeliegende Verhalten bleiben unklar (vgl. KAUFFELD, 2001, S.63f).

Neben einer Vielzahl von Fragebögen zu Denk-, Lern-, Problemlöse und Verhaltensstilen bilden die Fragebögen zu Teamrollen eine große Gruppe innerhalb der strukturanalytischen Verfahren. Auf einen dieser Fragebögen wird – auch im Hinblick auf die Relevanz für die vorliegende Arbeit – im Rahmen des nachfolgenden Kapitels näher eingegangen werden.

2.5 Rollen und Rollenmodelle

Die Bildung von Gruppen, insbesondere die von Arbeitsgruppen, aber auch die von Lerngruppen, erfolgt üblicherweise, weil die Komplexität der jeweils zu bewältigenden Aufgabe die Fähigkeiten einer einzelnen Person übersteigt und ein aufeinander abgestimmtes Zusammenspielen mehrerer Personen mit zum Teil unterschiedlichen Fähigkeiten (bzw. unterschiedlichen Kenntnissen) erfordert (s. Kap.2). Insbesondere erfolgt eine Aufteilung der anfallenden Arbeiten, sodass einzelne Gruppenmitglieder Teilaufgaben bearbeiten und die Ergebnisse letztlich geeignet zu einem Gesamtergebnis zusammengefügt werden. Die Arbeit ist dabei so zu organisieren, dass einerseits eine größtmögliche Effizienz erreicht wird und andererseits auch die Motivation der Gruppenmitglieder nicht beeinträchtigt wird. Dies bedeutet für die Arbeitsteilung eine höchstmögliche Aufgabenspezialisierung innerhalb der Gruppe unter Erhaltung des Interesses aller Mitglieder an ihrer Arbeit und damit verbunden deren Zufriedenheit.

Interpretiert man eine Rolle im Sinne einer solchen Arbeitsteilung, so lässt sich das Vorhandensein mehrerer Rollen in einer (Arbeits-)Gruppe durchaus begründen, denn *„für die Koordination und Differenzierung von Handlungen werden unterschiedliche Rollen benötigt, die von den Teammitgliedern übernommen werden"*

(Kriz & Nöbauer, 2002, S.47ff). Ähnlich formuliert es auch Mann: *„Die Spezialisierung der Rollen geschieht deshalb, weil die Lösung der alltäglichen Probleme eine kooperative Funktionsdifferenzierung innerhalb der Gruppe notwendig macht"* (Mann, 2001, S.59).

Welcher Art diese Differenzierung ist und welche Rollen daraus resultieren ist Gegenstand zahlreicher Forschungsprojekte mit durchaus unterschiedlichen Sichtweisen. Dementsprechend unterschiedlich sind auch die resultierenden Rollenmodelle, von denen nachfolgend die gängigsten Vertreter charakterisiert werden.

2.5.1 Bales und Slater

Robert F. Bales und Philip E. Slater haben in den 1950er Jahren das Rollenverhalten in Kleingruppen (vgl. Hare et al., 1955, S.610ff) untersucht und sich Fragestellungen zur Wahrnehmung unterschiedlicher Rollen durch die Gruppenmitglieder und zu möglichen Zusammenhängen zwischen einer Rolle und dem Verhalten eines Rollenträgers zugewandt. Dabei sehen sie das Konzept der Rolle in einer strategischen Position zwischen Psychologie und Soziologe und definieren Rollen als zusammenhängende, einheitliche Systeme zwischenmenschlicher Verhaltensweisen: *„We might define a role as a more or less coherent and unified system of items of interpersonal behaviour"* (Hare et al., 1955, S.610).

Bales und Slater haben 20 Kleingruppen (mit drei bis sieben Mitgliedern) in insgesamt 80 Gruppensitzungen bei der Diskussion und Entscheidungsfindung von fiktiven administrativen Problemen beobachtet und dabei sämtliche Äußerungen der Gruppenmitglieder protokolliert und anhand einer gegebenen Kategorisierung (*Bales' set of interaction categories*; vgl. Bales, 1950, S.8ff; Hare, 1962, S.66ff) klassifiziert. Die Auswertung dieser Daten mittels gängiger statistischer Verfahren führt unter anderem zur Identifizierung von zwei Rollen, dem *Idea man* und dem *Best-liked man*. Der Aufgabenführer einer Gruppe (*Idea man*) fungiert als Ideenlieferant für die Gruppe und leitet in der Regel solche Interaktionen ein, die letztlich zur Problemlösung führen. Der *Best-liked man* ist eine Art sozio-emotionaler Spezialist, der stets bemüht ist, die Moral innerhalb der Gruppe aufrecht zu erhalten und durch Konflikte verursachte Unterbrechungen der Gruppenarbeit zu verhindern. Weitere Rollen wurden von Bales und Slater nicht identifiziert.

2.5.2 Belbin

Im Jahr 1969 begann Bebin eine insgesamt neunjährige Studie zu Teamrollen und Teams, die auf der Analyse und dem anschließenden Vergleich von Persönlichkeitsmerkmalen von Mitgliedern erfolgreicher und weniger erfolgreicher Teams basiert. Die quantitative Messung der Persönlichkeitsmerkmale beruht auf den Ergebnissen des 16PF, einem von Raymond B. Cattell (1905-1996, britisch-amerikanischer Persönlichkeitspsychologe) entwickelten Persönlichkeitstest: Mittels eines Fragebogen werden 16 Primärdimensionen der Erwachsenenpersönlichkeit ermittelt, aus

denen wiederum fünf Globalfaktoren (namentlich Extraversion, Unabhängigkeit, Ängstlichkeit, Selbstkontrolle und Unnachgiebigkeit) abgeleitet werden. Die quantitative Messung des Erfolgs wird unter objektiven Bedingungen im Rahmen einer einwöchigen Managerschulung anhand eines Business-Computerspiels durchgeführt. Parallel dazu wurden von jeder Gruppe durch geschulte Beobachter Prozessanalysedaten mittels eines auf IPA basierenden Verfahrens erhoben, anhand derer von jeder Gruppe ein qualitativer Bericht bzgl. ihrer Zusammenarbeit erstellt werden konnte (vgl. KAUFFELD, 2001, S.82ff).

Auf dieser Grundlage identifizierte Belbin zunächst acht Teamrollen (vom Macher über den Erfinder bis hin zum Teamarbeiter). Dieses Rollenmodell wurde später vom Autor selbst um eine weitere Rolle, den Spezialisten, ergänzt: *„Belbin (1981) claimed to have identified eight team roles as a result of his original research using management teams playing management games. In his later book, published in 1993, he renamed some of the team roles and added a ninth role, specialist, as a result of further research. So, Belbin's latest work describes nine possible team roles."* (PARK & BANG, 2002, S.3). Eine Charakterisierung aller von Belbin identifizierten Rollen ist der Tabelle 2.1 zu entnehmen.

Die Bestimmung der Teamrollen ist mittlerweise anhand eines von Belbin entwickelten Fragebogens, dem *Belbins Team-Role Self-Perception Inventory* (BTRSPI), durchführbar. Jede Fragestellung stellt für jede Teamrolle ein Antwort-Item bereit. Durch Addition der Punkte über die Fragen ergeben sich Tendenzen zu den einzelnen Teamrollen. Die höchste Punktzahl repräsentiert die sog. primäre Teamrolle, die zweithöchste Punktzahl beschreibt die sog. Backup-Rolle und die niedrigsten Punktzahlen weisen auf mögliche Defizite bezüglich der zugehörigen Teamrollen hin.

Tab. 2.1: Teamrollen nach Belbin

Teamrolle	**Stärken**	**Schwächen**
Neuerer/Erfinder (Plant; PL)	Kreativ, phantasievoll, unorthodox; löst schwierige Probleme	Zu beschäftigt, um wirksam zu kommunizieren
Wegbereiter (Resource Investigator; RI)	Extrovertiert, enthusiastisch, kommunikativ; erkundet Gelegenheiten	Über-optimistisch; verliert nach erstem Enthusiasmus schnell das Interesse
Vorsitzender (Coordinator; CO)	Reif, zuversichtlich, ein guter Vorsitzender; klärt Ziele, fördert Entscheidungsfindung, kann gut delegieren	Kann als manipulativ angesehen werden, lädt persönliche Arbeiten auf andere ab

Fortsetzung auf nächster Seite

Tab. 2.1: Teamrollen nach Belbin *Forts.*

Teamrolle	**Stärken**	**Schwächen**
Macher (Shaper; SH)	Herausfordernd, dynamisch, entfaltet sich unter Druck; Antrieb und Mut zur Bewältigung von Hindernissen	Anfällig für Provokationen, verletzt die Gefühle anderer
Beobachter (Monitor Evaluator; ME)	Nüchtern, strategisch, kritisch; sieht alle Möglichkeiten, urteilt treffsicher	Mangel an Schwung und Fähigkeit, andere zu begeistern
Teamarbeiter (Team Worker; TW)	Kooperativ, sanft, scharfsinnig, diplomatisch; hört anderen zu, verhindert Reibungen	Unentschlossen in Krisensituationen
Umsetzer (Implementer; IMP)	Diszipliniert, zuverlässig, konservativ, effizient; wandelt Ideen in die Praxis um	Etwas inflexibel; stellt sich nur langsam auf neue Möglichkeiten um
Perfektionist (Completer-Finisher; CF)	Sorgfältig, gewissenhaft, besorgt; macht Fehler und Unterlassungen ausfindig; termintreu	Neigt zu übermäßiger Sorge; delegiert nur widerwillig
Spezialist (Specialist; SP)	Zielstrebig, motiviert, engagiert; stellt Wissen und Fähigkeiten in seinem Fachgebiet zur Verfügung	Leistet Beiträge nur aus einem begrenzten Bereich; ergeht sich in technischen Details

(Park & Bang, 2002, S.4)

Aus seinem Rollenmodell leitet Belbin weiterhin fünf Prinzipien zur Bildung eines effektiven Teams ab (vgl. Kauffeld, 2001, S.87):

1. Jedes Teammitglied vertritt eine funktionale Rolle (in Form von beruflichem bzw. technischem Wissen) und eine Teamrolle.

2. Jedes Team braucht ein ausgewogenes Verhältnis aus funktionalen Rollen und Teamrollen. Diese *Teambalance* ist abhängig von den jeweilgen Gruppenzielen.

3. Die Leistungsfähigkeit eines Teams ist abhängig davon, wie gut sich die Teammitglieder selbst einschätzen und sich – im Hinblick auf Fachwissen und Teamrollen – an das Team anpassen.

4. Teammitglieder nehmen unterschiedliche Teamrollen – abhängig von ihren Neigungen und Fähigkeiten – unterschiedlich gut wahr.

5. Eine optimale Nutzung seiner Ressourcen durch ein Team kann nur dann erfolgen, wenn die Teamrollen mit ausreichender Bandbreite und Ausgewogenheit vorhanden sind.

So einleuchtend diese Prinzipien und die zugrundeliegenden Teamrollen sein mögen, so bleiben dennoch berechtigte Fragen unbeantwortet. Insbesondere fehlt ein geeigneter Maßstab für die geforderte Teambalance oder anders ausgedrückt: Wann ist ein Team ausgewogen besetzt? Und welchen Einfluss haben Gruppenziel beziehungsweise Gruppenaufgabe auf die Teambalance? Dass alle Teamrollen in irgendeiner Form ausreichende Ausprägung bei angemessener Verteilung besitzen, wird auch von anderen Autoren im Zusammenhang mit anderen Rollenmodellen gefordert (vgl. Kriz & Nöbauer, 2002, S.50; Spencer & Pruss, 1995, S.58; Eunson, 1990, S.427f). Weiterhin muss sich Belbin der Kritik stellen, dass sein Rollenmodell in der Hauptsache auf empirisch gewonnenen Daten beruht und objektiv nachvollziehbare Belege unerwähnt bleiben: *„Die Begeisterung der Berater für den BTRSPI wird von den Psychometrikern nicht geteilt. Broucek und Randell (1996) beklagen, dass die neun Teamrollen auf unterschiedlichste Weise und anhand von inkonsistenten Kriterien identifiziert wurden. Sie konstatieren einen Mangel an statistischen Beweisen für theoretische Inhalte und sehen die Teamrollen nur durch anekdotische Beschreibungen belegt"* (Kauffeld, 2001, S.87f). Trotz dieser Kritik kommen das Modell von Belbin und der zugehörige Fragebogen (BTRSPI) in der Praxis häufig zur Anwendung (vgl. Kauffeld, 2001, S.87).

2.5.3 Margerison und McCann

Aus der Fragestellung *„Was machen erfolgreiche Führungskräfte und Teams eigentlich anders als andere?"* begannen Charles Margerison und Dick McCann 1982 mit der Entwicklung eines speziell für Arbeitsplätze in Industrie und Wirtschaft anwendbaren psychologischen Instruments. Im Rahmen ihrer Untersuchungen interviewten sie *„tausende von Führungskräften aller Managementebenen, Team- und Projektleiter sowie Teammitglieder. Sie sprachen mit Teams, die offensichtlich erfolgreich – oder offenbar erfolglos arbeiteten"* Wagner (2005). Als Ergebnis identifizierten sie neun Schlüsselfunktionen der Teamarbeit, die unabhängig vom jeweiligen Arbeitskontext gültig sind (vgl. Margerison et al., 1995, S.13f; Kauffeld, 2001, S.78f; Wagner, 2005). Eine kurze Charakteristik ist der nachfolgenden Tabelle zu entnehmen.

Tab. 2.2: Schlüsselfunktionen der Teamarbeit nach McCann & Margerison

Schlüsselfunktion	**Beschreibung**
Beraten (Advising)	Informationen beschaffen und verbreiten
Innovationen schaffen (Innovating)	Ausdenken neuer Ideen und Lösungsansätze
Fördern (Promoting)	Gezielte Weitergabe neuer Ideen sowie Erkunden und Präsentieren von Möglichkeiten

Fortsetzung auf nächster Seite

Tab. 2.2: Schlüsselfunktionen der Teamarbeit nach McCann & Margerison *Forts.*

Schlüsselfunktion	**Beschreibung**
Entwickeln (Developing)	Planung und Auswahl von Verfahren zur praktischen Umsetzung von Ideen in Produkte und Dienstleistungen
Organisieren (Organizing)	Planerische Vorbereitung und Schaffung geeigneter Rahmenbedingungen für Umsetzung
Produzieren (Producing)	Erstellung der Produkte bzw. Dienstleistungen gemäß gegebener Pläne und Termine
Prüfen (Inspecting)	Kontrolle der Ergebnisse mit dem Ziel, gesetzte Qualitätsstandards einzuhalten
Instandhalten (Maintaining)	Unterstützungsfunktion zur Bereitstellung einer funktionierenden Infrastruktur
Verbinden (Linking)	Koordination und Integration aller anderen Arbeitsfunktionen

(Margerison et al., 1995, S.13f)

Desweiteren versuchten McCann und Margerison herauszufinden, warum manche Arbeiten von einigen Personen bevorzugt und von anderen abgelehnt werden. Nach umfangreichen Untersuchungen auf Basis der Jungschen Typenlehre (vgl. Schmid & Caspari, 1998; Dorsch, 1982, S.705) definieren die Autoren vier Schlüsselbereiche, die für das verschiedenartige Verhalten von Menschen bei der Arbeit wesentlich sind, als da wären

- die bevorzugte Art, mit anderen Menschen zu kommunizieren (extrovertiert vs. introvertiert),
- die bevorzugte Art, Informationen zu sammeln und zu nutzen (faktenorientiert vs. kreativ),
- die bevorzugte Art, Entscheidungen zu treffen (analytisch vs. gefühlsorientiert) und
- die bevorzugte Art, sich selbst und andere zu organisieren (planvoll vs. flexibel).

Die Messung der Ausprägungen dieser vier Arbeitspräferenzen erfolgt anhand des *Team Management Fragebogens*: *„Die Präferenzen oder Rollen innerhalb eines Teams werden durch einen 60 Punkte umfassenden Fragebogen ermittelt, bei dem das Teammitglied Wortpaare entsprechend seiner Neigungen und Einstellungen gewichtet"* (Kauffeld, 2001, S.81).

Beide Ansätze wurden letztlich miteinander verknüpft, so dass Beziehungen zwischen Arbeitsfunktionen und Persönlichkeitsmerkmalen hergestellt werden konnten. Das Ergebnis notieren die Autoren in Form von acht Teamrollen, die sie *Informierter Berater*, *Kreativer Innovator*, *Entdeckender Promoter*, *Auswählender*

Entwickler, *Zielstrebiger Organisator*, *Systematischer Umsetzer*, *Sorgfältiger Überwacher* und *Unterstützender Stabilisator* nennen. Das Ergebnis wird durch das *Team Management Rad* bildlich beschrieben (s.Abb.2.6).

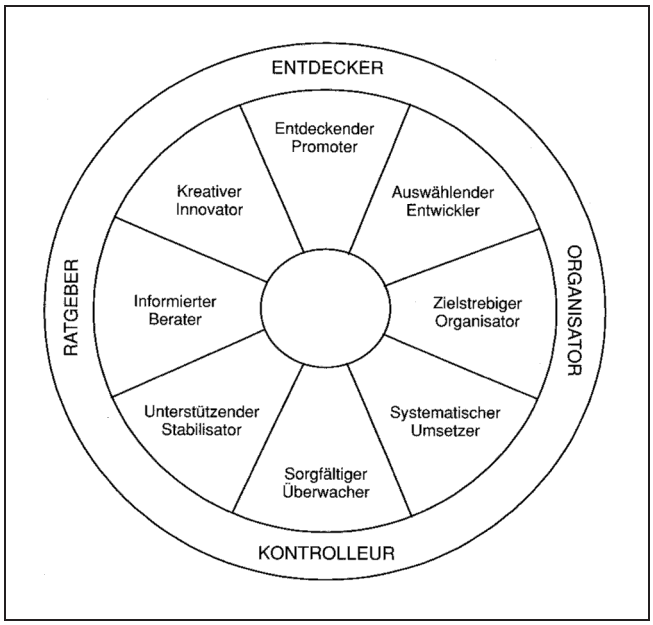

Abb. 2.6: Das Team Management Rad nach MARGERISON ET AL. (KAUFFELD, 2001, S.81)

Eine detailierte Beschreibung sämtlicher Teamrollen ist bei Bedarf dem Anhang B.1 entnehmbar.

2.5.4 Eunson's Ansatz zur Rollenanalyse

In seinem Buch über verschiedene Aspekte menschlichen Verhaltens im Rahmen der Betriebspsychologie stellt EUNSON (1990) ein nach verschiedenen Rollentypen kategorisiertes Rollenmodell vor. Zur besseren Analyse von Gruppen schlägt der Autor vor, *„zwischen drei Typen von Rollen zu unterscheiden, die für gewöhnlich wirksam sind – Aufgabenrollen, sozio-emotionale Rollen und destruktive Rollen"* (EUNSON, 1990, S.247):

- **Aufgabenrollen**:
  Die Wahrnehmung der Aufgabenrollen ist notwendig im Sinne der Aufgabenbearbeitung durch die Gruppe. Diese Rollen sind demnach durch fachspezifische beziehungsweise aufgabenbezogene Tätigkeiten spezifiziert. Eine kurze Charakteristik aller Aufgabenrollen ist dem Anhang (Kap.B.2.1, Tab.B.1) zu entnehmen.

- **Sozio-emotionale Rollen**:
  Die Wahrnehmung dieser Rollen dient dem Aufbau und der Stärkung konstruktiver zwischenmenschlicher Beziehungen sowie der konstruktiven Kom-

munikation innerhalb der Gruppe. Diese Unterstützungsrollen sind zusätzlich zu den o.g. Aufgabenrollen durch die Gruppenmitglieder wahrzunehmen. Skizzierungen der einzelnen sozio-emotionalen Rollen finden sich im Anhang (Kap.B.2.2, Tab.B.2).

- **Zerstörerische Rollen**:
  Die Rollen dieses Typs sind – im Gegensatz zu denen der beiden vorigen Kategorien – schädlich für die Gruppenarbeit und daher möglichst zu vermeiden. Aus diesem Grund enthalten die zugehörigen Beschreibungen (s. Kap.B.2.3, Tab.B.3) teilweise auch Strategien, wie mit den Trägern dieser Rollen zu verfahren ist.

So anschaulich auch jede der genannten Teamrollen beschrieben ist und so einleuchtend ihr Nutzen oder Schaden für die Gruppenarbeit sein können, so sehr fehlt letztlich die wissenschaftliche Begründung des Zustandekommens dieses Rollenmodells: Man gewinnt den Eindruck, dass sich Eunson (1990) aus mehreren Quellen bedient hat, denn es lassen sich bei genauerem Hinsehen alle Eigenschaften auch in Belbin's Teamrollen (s. Kap.2.5.2) wiederfinden. So ist Belbin's Perfektionist in Eunson's Rollenaufzählung sowohl im Geschäftsordnungspraktiker als auch anteilig im Detailversessenen oder im Definierer erkennbar. Desweiteren fehlt eine Operationalisierung der Rollen gänzlich, das heißt der Autor benennt zwar mögliche Rollen und skizziert ihre Eigenschaften, jedoch liefert er kein Verfahren, um für Mitglieder einer Gruppe ein Rollenprofil zu bestimmen. Trotz dieser Kritik mag Eunson's Modell in der betriebspsychologischen Praxis durchaus Anwendung finden, um Gruppenstrukturen zu beschreiben und Problemsituationen anhand einer gegebenen Struktur genauer zu betrachten.

2.5.5 Spencer und Pruss

Wie andere Autoren zuvor (s. Kap.2.5.2 und 2.5.3) gehen auch Spencer & Pruss (1995) von der Annahme aus, dass Mitglieder erfolgreicher Teams bestimmte Funktionen wahrnehmen müssen, das heißt die Teammitglieder müssen bestimmte Rollen einnehmen. Jedoch kritisieren sie, dass bei der Beschreibung von Rollen und Rollenmodellen oftmals nicht zwischen Teamrollen und Persönlichkeitstypen differenziert wird: *„Es muss gesagt werden, dass unserer Meinung nach eine unnötige und wenig hilfreiche Verwischung zwischen den Definitionen der Teamrollen und der individuellen Persönlichkeitstypen stattgefunden hat, indem bestimmte Persönlichkeitstypen bestimmten Funktionen zugewiesen wurden"* (Spencer & Pruss, 1995, S.58).

Spencer & Pruss (1995) beschreiben ihre Teamrollen daher nicht durch psychologische Eigenschaften, sondern berücksichtigen vielmehr Fähigkeiten in den Bereichen Kommunikation und Einflussnahme. Unter diesem Aspekt analysierten die Autoren im Rahmen ihrer Tätigkeit als Unternehmensberater in Sachen Teamarbeit und Personalmanagement unter anderem Personalakten, Bewerbungen und Lebensläufe. Als Ergebnis präsentieren sie ein Modell aus zehn Teamrollen, die im

Hinblick auf effiziente Teamarbeit einzunehmen sind. Jede Teamrolle besitzt neben einer funktionalen Beschreibung auch eine Charakterisierung der wesentlichen Persönlichkeitsmerkmale typischer Vertreter der jeweiligen Rolle.

Tab. 2.3: Teamrollen nach Spencer & Pruss

Teamrolle	**Beschreibung**	**Merkmale**
Visionär	Überblickt den Teamauftrag in seiner Gesamtheit; schwebt über den Wolken und greift nach den Sternen	Einfluss durch Überzeugung; optimitisch, offen und ehrlich; kein Interesse an Details; ungeduldig; künstlerisch
Pragmatiker	Gegenpol zum Visinonär; macht auf Rahmenbedingungen und Probleme aufmerksam; zeigt praktikable Lösungsansätze	Einfluss durch Druck; realistisch, zynisch und skeptisch; Teamspieler; mathematischer oder naturwissenschaftlicher Hintergrund
Entdecker	Beschaffung von Informationen, Materialien, Unterstützung etc. von außerhalb des Teams; Teambotschafter	gesellig, neugierig, ehrgeizig; hohe Selbstmotivation; kommunikationsfähig
Herausforderer	Stellt Bestehendes in Frage; sorgt für Weiterverfolgung der Teamziele	desillusioniert, zynisch mit optimistischer Grundhaltung; Generalist statt Spezialist
Unparteiischer	Außenstehender; Berater	Flexibel, neutral, optimistisch und enthusiastisch; entschlussfreudig, humorvoll, glaubwürdig
Friedensstifter	Bereinigt Meinungsverschiedenheiten und Konflikte	kommunikationsfähig, selbstbewusst, objektiv; denkt logisch; engagiert, charakterfest
Arbeitstier	Erledigt die anfallende Arbeit	aufgabenorientiert statt personenorientiert; braucht Anerkennung; kaum kreativ; lebt von Regeln und Vorschriften; nicht ehrgeizig
Trainer	Stärkt Teammoral; sorgt für neuen Antrieb; treibende Kraft	Taktiker; ehemaliger Visionär; glaubwürdig; reife Persönlichkeit mit viel Erfahrung
Bibliothekar	Zeichnet Teamaktivitäten auf; Archiv des Teams	Zurückhaltend, detailverliebt, fleissig und gewissenhaft; intolerant und leicht erregbar; guter Präsentator von Fakten
Beichtvater	Jemand, bei dem man seine Probleme abladen kann	Gesellig und ehrlich, aber oberflächlich; vertrauenswürdig; guter Universitätsabschluss; mobil

(Spencer & Pruss, 1995, S.60ff)

Zu den in der Tabelle 2.3 kurz charakterisierten zehn Teamrollen von Spencer & Pruss (1995) ist eine komplette, ausführliche Beschreibung dem Anhang B.3 entnehmbar.

In der Folge haben die Autoren einen Fragebogen entwickelt, der 150 Aussagen enthält, die sich nicht ausschließlich auf die Arbeitssituation beziehen. *„Der ... Fragebogen ist darauf angelegt herauszufinden, zu welchen der oben beschriebenen Klassifikationen jemand tendiert; er wird den Teamleitern und anderen Teammitgliedern einen Hinweis auf ihre Stärken und Schwächen geben, dazu einen gewisen Eindruck des Teamgleichgewichts innerhalb ihrer jeweiligen Gruppierungen vermitteln"* (Spencer & Pruss, 1995, S.75). Das Ergebnis eines von einem Teammitglied ausgefüllten Fragebogens lässt sich in Form eines Rollenprofils anschaulich darstellen. Die nachfolgende Abbildung 2.7 illustriert anhand eines Beispiels solch ein Rollenprofil.

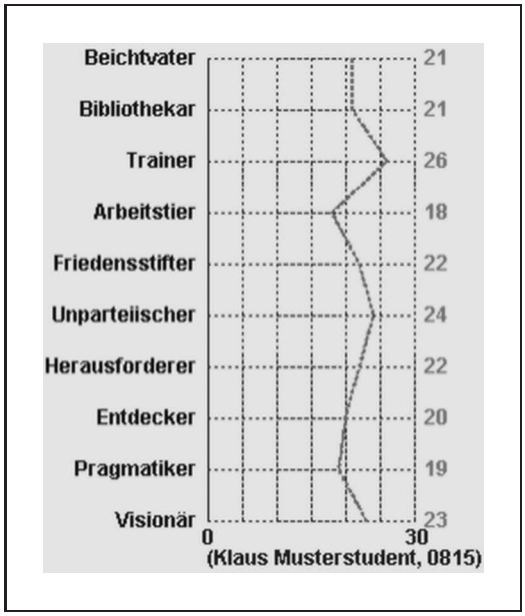

Abb. 2.7: Beispiel eines Rollenprofils nach Spencer & Pruss

Die einzelnen Rollenausprägungen lassen sich sehr leicht dem in der Abbildung dargestellten Diagramm entnehmen und entsprechend der Hinweise in Spencer & Pruss (1995) interpretieren. Insbesondere liefert der Teamfragebogen nicht nur Hinweise auf geeignete Teamzusammensetzungen oder durchzuführende Teamentwicklungsmaßnahmen, sondern lässt sich auch zur Operationalisierung der Teamrollen anwenden. Der komplette Fragebogen sowie Hinweise zur Auswertung sind im Anhang B.3.2 zu finden.

Insgesamt betrachtet liefern Rollenmodelle wie das von Spencer & Pruss (1995) einen pragmatischen Ansatz zur Beschreibung von Gruppenstrukturen im Sinne zu erfüllender beziehungsweise wahrzunehmender Teamfunktionen und somit auch hilfreiche Informationen für weitere Teamentwicklungsmaßnahmen. Speziell die Möglichkeit der Operationalisierung der Teamrollen mittels Fragebogen macht das hier vorgestellte Rollenmodell für weitere Forschungen interessant.

2.5.6 Rollen in der Software-Entwicklung

Das *V-Modell XT* ist ein Modell zur Durchführung von IT-Projekten, das seit dem 4.11.2004 für Behörden der Bundesverwaltung verbindlich ist. Das V-Modell XT definiert die Aktivitäten (Tätigkeiten) und Produkte (Ergebnisse), die während der Entwicklung von Systemen durchzuführen beziehungsweise zu erstellen sind. Darüber hinaus legt es die Verantwortlichkeiten jedes Projektbeteiligten fest. Das V-Modell regelt also detailliert, wer wann was in einem Projekt zu tun hat. Die Regelung sämtlicher Verantwortlichkeiten erfolgt anhand eines Rollenmodells, das aus 30 Einzelrollen (vom Akquisiteur bis hin zum Technischen Autor) besteht: *„Eine Rolle ist die Beschreibung einer Menge von Aufgaben und Verantwortlichkeiten im Rahmen eines Projekts und einer Organisation. Durch die Festlegung von Rollen wird die Unabhängigkeit des V-Modells von organisatorischen und projektspezifischen Rahmenbedingungen erreicht. Die Zuordnung von Organisationseinheiten und Personen zu den Rollen erfolgt zu Beginn eines Projekts"* (von Hagen, 2005, S.36).

Im Gegensatz zu eher psychologisch basierten Rollenmodellen (vgl. Hare et al., 1955; Kauffeld, 2001, S.82ff; Eunson, 1990) stellen die Rollen des V-Modells ausschließlich funktionale Rollen im Sinne wahrzunehmender Aufgaben dar. Hinweise auf wünschenswerte Charaktereigenschaften oder andere psychologische Merkmale möglicher Rollenträger fehlen gänzlich. Im Hinblick auf die Zusammensetzung von Teams und deren Effizienz besitzt dieses Rollenmodell also zunächst keine Relevanz, es kann jedoch durchaus als nahezu vollständige Übersicht über diejenigen Funktionen dienen, die in IT-Projekten zur erfolgreichen Zielerreichung als notwendig erachtet werden. Somit kann es im Rahmen der vorliegenden Arbeit durchaus als Hilfsmittel zur Überprüfung der Vollständigkeit anderer Rollenmodelle und der in ihnen definierten Teamfunktionen herhalten. Da eine ausführliche Beschreibung aller Rollen des V-Modell XT an dieser Stelle nicht hilfreich ist, sei für weitere Informationen auf KBSt (2004) verwiesen. Insbesondere ist das V-Modell XT für den Einsatz in langfristig orientierten Großprojekten konzipiert, wohingegen in der vorliegenden Arbeit Kleingruppenprojekte mit einem eher geringen zeitlichen Horizont betrachtet werden.

2.5.7 Rollen in Lernsituationen

Steht nicht die aufgaben- oder produktorientierte Zusammenarbeit im Vordergrund, sondern erfolgt die Zusammenarbeit mit dem Ziel, Wissen (über ein bestimmtes Fachgebiet oder einen speziellen Problembereich) zu erwerben, so sprechen wir von einer Lerngruppe: *„Lerngruppen erstellen zwar auch meist ein Produkt, das eigentliche Ziel ist jedoch der Erkenntnisgewinn jedes Einzelnen. Während das Ergebnis das Mittel zur Erreichung des Ziels darstellt, ist die Entwicklung von Sozial-, Fach- und Methodenkompetenz Teil des Gruppenziels. Die Aufteilung der Aufgaben erfolgt so, dass Qualifizierungsdefizite möglichst abgebaut werden. Die Gruppenmitglieder sind intrinsisch motiviert, sie wollen das Ziel erreichen"* (Haake et al., 2004, S.203). Auch hier wird auf eine Differenzierung

der einzelnen Mitglieder der Lerngruppe hingewiesen, die für eine erfolgreiche Aufgabenbearbeitung unabdinglich ist.

Lernen mit dem Erwerb von Wissen gleichzusetzen stellt eine starke Vereinfachung dar und umfasst bei weitem nicht alle Aspekte des Lernens. So wird Lernen von Zimbardo & Gerrig (2000) wie folgt definiert: *„Wir können Lernen als einen Prozess definieren, der zu relativ stabilen Veränderungen im Verhalten oder im Verhaltenspotential führt und auf Erfahrung aufbaut. Lernen ist nicht direkt zu beobachten. Es muss aus den Veränderungen des beobachtbaren Verhaltens erschlossen werden"* (Zimbardo & Gerrig, 2000, S.623). Ergänzend zu dieser Definition wird Lernen auch beschrieben

- als Herstellen von Assoziationen zwischen Vorstellungen,
- als Herstellen von bedingten Reaktionen durch Assoziationen eines bedingten Reizes mit einem unbedingten, wodurch der bedingte die Fähigkeit erhält, die Reaktion auf den unbedingten auszulösen (klassisches Konditionieren),
- als Erwerben einer neuen instrumentellen Reaktion durch Auswahl und Zusammensetzung erfolgreicher Bewegungen unter Stimulus-Kontrolle, motiviert durch den Antrieb und von Verstärkungen abhängig (instrumentelles Konditionieren) sowie
- als Neuorganisation der Situation durch die Bildung von neuen Strukturen, also Umzentrierung oder Neugliederung des Lernmaterials durch Klassifizierung oder »cluster«-Bildung (vgl. Dorsch, 1982, S.382).

Unabhängig von den vorangegangenen Definitionen mag im Rahmen der vorliegenden Arbeit Lernen im Sinne eines individuellen Erwerbs von Wissen eine hinreichend genaue Definition darstellen. Wissen – als Grundlage eines jeden Lernprozesse – wird von Kuhlen et al. (2004) verstanden als *„im Bewusstsein verfügbare Kenntnisse über Gegenstände, Sachverhalte, Personen, Ereignisse, Methoden, Regeln etc. einschließlich des zugehörigen lebensweltlichen (historischen) Begründungszusammenhangs"* (Kuhlen et al., 2004, S.53f). Aus der Anwendung von Wissen wird Information generiert, die – nach erfolgter Übertragung, also im Rahmen eines kommunikativen Akts – auf der Empfängerseite ihrerseits zur Generierung neuen Wissens genutzt werden kann (vgl. Kuhlen et al., 2004, S.53f).

Das Lernen in der Gruppe beinhaltet somit den Austausch von individuellem Wissen (in Form von Information) zwischen den Gruppenmitgliedern und infolgedessen auch den Aufbau einer gemeinschaftlichen Wissensbasis. Dieser Austausch bedingt also den Transport von Wissen – in Teilen oder als Ganzes – zwischen einem Sender und einem Empfänger (bzw. *source* und *destination*; vgl. Gerbner, 1989, S.17f). Das Ziel des Lernens als Folge von Wissensaustausch besteht folglich darin, auf der Empfängerseite die oben genannten Qualifizierungsdefizite abzubauen. Die an diesem Vorgang Beteiligten werden in dieser Arbeit als *Wissensvermittler* und *Wissensempfänger* bezeichnet.

2.5.7.1 Der Wissensvermittler

Der Träger dieser Rolle verkörpert den Experten, der das notwendige fachliche Wissen besitzt, um eine gegebene Aufgabe erfolgreich zu bearbeiten (vgl. (Zimbardo & Gerrig, 2000, S.381f)). Darüber hinaus besitzt er auch die notwendigen Kompetenzen, um sein Wissen an (wesentlich) schwächere Gruppenmitglieder weitergeben zu können. Dieses Verständnis von Wissensvermttlung entspricht dem Ansatz des *Cognitive Apprenticeship*, der auf der Annahme beruht, *„dass die Fähigkeit von Lernenden, Probleme zu lösen, in dem Maße erleichtert wird, wie es gelingt, ihnen zu vermitteln, wie ein Experte ein anstehendes Problem löst"* (Konrad & Traub, 2005, S.34).

2.5.7.2 Der Wissensempfänger

Die Aufgabe dieses Gruppenmitglieds ist eingebettet in eine Lernsituation und besteht in der Lösung eines Problems beziehungsweise in der Bearbeitung einer Aufgabe verbunden mit dem Erwerb von Wissen. Der Wissensempfänger besitzt anfangs nur geringes Grundwissen über das zur Aufgabe zugehörige Themengebiet. Er wird dabei vom Wissensvermittler unterstützt und angeleitet. Idealerweise besitzt der Wissensempfänger am Ende des Lernprozesses so viel Wissen über das Aufgabengebiet, dass auch er fortan als Wissensvermittler fungieren kann. Abhängig vom verwendeten lerntheoretischen Ansatz finden sich diese beiden Rollen in unterschiedlichen Ausprägungen wieder. Dies trifft insbesondere auf den Wissensvermittler zu, der – je nach Lerntheorie – als Lehrer (Behaviourismus) , Tutor (Kognitivismus) oder Coach (Konstruktivismus) verstanden wird.

2.5.7.3 Lerntheoretische Differenzierung

Je nach zugrundeliegender Lerntheorie wird die Vermittlung von Wissen unterschiedlich charakterisiert. Im Folgenden wird dies unter dem Gesichtspunkt von Behaviourismus, Kognitivismus und Konstruktivismus näher charakterisiert.

2.5.7.3.1 Wissensvermittlung im Behaviourismus Im Sinne des Behaviourismus wird Wissen verstanden als eine korrekte Input-Output-Relation, das heißt auf einen Reiz (*Stimulus*) erfolgt eine bestimmte Antwort (*Response*). Bei dieser Sichtweise wird der Lernende reduziert auf eine *Black box*, die dem Lernen zugrundeliegenden internen Prozesse werden nicht beachtet. Der Lehrer als Wissensvermittler ist eine Autorität mit dem Ziel, im Wissensempfänger die gewünschten Stimulus-Response-Relationen zu bilden, ggf. unter Einbeziehung von Belohnungen und Bestrafungen. In dieser Sichtweise wird Wissen also lediglich im Wissensempfänger abgelegt. Bezogen auf die eingangs geschilderte Gruppensituation ist dieses Modell kaum anwendbar, da keine gemeinschaftliche Konstruktion von Wissen stattfindet.

2.5.7.3.2 Wissensvermittlung im Kognitivismus In der Theorie des Kognitivismus wird der Lernende als ein Individuum verstanden, das nicht ausschließlich durch äußere Reize steuerbar ist, sondern externe Stimuli aktiv und selbständig verarbeitet. Das Lernen ist nunmehr ein Informationsverarbeitungsprozess, der stets im Austausch mit der Umwelt stattfindet. Hieraus hat sich das Konzept des Entdeckenden Lernens entwickelt: *„Nicht mehr das Wissen um die richtige Lösung steht im Zentrum eines Lehr-/Lernprozesses, sondern der Aufbau des Verständnisses für ein Problem und damit der Aufbau von Problemlösekompetenz, die es dem Lernenden ermöglicht, sich die Lösung eines Problems selbstständig zu erarbeiten"* (Dittler, 2003b, S.24). Der Wissensvermittler wird in dieser Lernsituation weniger als Autorität, sondern vielmehr als ein Tutor aufgefasst, der den Lernenden und dessen Lernprozess beobachtet und unterstützt.

2.5.7.3.3 Wissensvermittlung im Konstruktivismus Lernen im Konstruktivismus ist ein aktiver Prozess, der nur über die aktive Beteiligung des Lernenden möglich ist. Im Gegensatz zum Kognitivismus wird der Lernende jedoch als ein informationell geschlossenes System aufgefasst, in welchem Wissen auf der Basis von individueller Wahrnehmung, Interpretation und Konstruktion generiert wird. Interne Vorgänge finden somit weitaus mehr Beachtung als Wechselwirkungen mit der Umwelt: *„Ohne individuellen Erfahrungs- und Wissenshintergrund und eigene Interpretationen finden im Prinzip keine kognitiven Prozesse statt"* (Dittler, 2003b, S.127).

In dem Maße, wie der Lernende in den Vordergrund rückt, zieht sich der Wissensvermittler zurück und übernimmt die Rolle eines Coach[3]. Die Lernsituation, in welcher sich der Coach und sein Schützling befinden, ist geprägt von Kooperation. Durch die Konstruktion von Wissen wird der Lernende nicht nur in die Lage versetzt, richtige Antworten zu finden oder die richtigen Methoden zur Antwortfindung auszuwählen und anzuwenden, sondern auch komplexe Situationen zu meistern.

Der bereits eingangs erwähnte Ansatz des *Cognitive apprenticeship* taucht in diesem Zusammenhang erneut auf: *„Die Betonung beim Konzept des »cognitive apprenticeship« liegt auf einem Lernen, das eingebettet ist in einen sozialen Kontext, an dem Meister und Lehrling gleichermaßen teilhaben. ... Elemente der Lehrlingsausbildung ... sind die Beobachtung des Meisters durch den Lehrling mit dem Ziel, ein Modell zu bilden (modeling), der eigene Übungsprozess des Lehrlings mit Beratung durch den Meister (coaching) und die allmähliche Rücknahme der tutoriellen Aktivität (fading)"* (Schulmeister, 1997, S.81).

Dort, wo Gruppenlernen sinnvoll stattfinden kann, nehmen die Mitglieder von Lerngruppen üblicherweise sowohl die Rolle des Wissensvermittlers als auch die

[3] *„Der Begriff »Coach« leitet sich aus dem Wort »Kutsche« ab und kommt ursprünglich aus dem Ungarischen. 1855 verwendet man Coach erstmals in England und den USA im Sport, wo ein Coach der Trainer eines Sportteams ist. Der heutige Begriff des Coaching wurde aus dem Hochleistungssport übernommen,wo er für die persönliche und umfassende Betreuung eines Sportlers steht"* (Haake et al., 2004, S.219).

des Wissensempfängers wahr. Jedes Gruppenmitglied ist Lehrer, wenn vertiefte Kenntnisse aus dem jeweiligen Problembereich vorhanden sind, und Lernender, wenn Wissens- oder Qualifikationsdefizite in eben diesem Bereich bestehen – und zwar unabhängig von einer aufgrund einer Lerntheorie favorisierten Ausprägung dieser beiden Rollen.

2.6 Zusammenfassung

Gruppen bilden sich in der Regel, um dem Einzelnen die Erreichung bestimmter Ziele zu vereinfachen oder gar erst zu ermöglichen. Dies setzt voraus, dass die Mitglieder einer solchen Gruppe im Hinblick auf eine erfolgreiche Aufgabenbewältigung geeignet interagieren und ihre Aktivitäten aufeinander abstimmen. Zwar wird es immer wieder herausragende Leistungen Einzelner geben, aber zweifelsohne hat sich Gruppenarbeit in weiten Teilen als überlegen gegenüber individuellen Leistungen gezeigt. Dies betrifft Gruppenarbeit im engeren Sinne (im Kontext betrieblicher Arbeitsprozesse) genauso wie das Lernen in der Gruppe (vgl. Teufel et al., 1995, S.3,S.43ff; Kauffeld, 2001, S.4ff; Pfister & Wessner, 2001a, S.7). Zu beachten ist, dass effiziente Gruppenarbeit stets die Wahrnehmung unterschiedlicher Funktionen durch die Gruppenmitglieder erfordert: *„Grundsätzlich sind verschiedene Rollen in einem Team notwendig, und Personen können durch die Übernahme von unterschiedlichen Rollen einen wichtigen Beitrag zur Teamarbeit leisten"* (Kriz & Nöbauer, 2002, S.50). Mit Rollenmodellen werden die notwendigen Teamfunktionen beschrieben, in der Regel verbunden mit dem Hinweis, dass alle Rollen im Hinblick auf eine Zielerreichung zu besetzen sind: *„The Types of Work model provides a practical way to help managers and team members understand individual work interests and team needs. All areas must be covered for each team and organisation to survive and succeed"* (Margerison, 2005, S.23). Meist sind in den Charakterisierungen der einzelnen Rollen auch die Anforderungen an den jeweiligen Rollenträger enthalten. Dadurch entsteht oftmals eine Unschärfe im Rollenbegriff, verursacht durch die gleichzeitige Betrachtung von Rollen als Teamfunktionen und Persönlichkeitsmerkmalen (vgl. Spencer & Pruss, 1995, S.58).

Eine Rolle als solche stellt – unabhängig von ihrer jeweiligen Ausprägung – zunächst eine Erwartungshaltung dar, deren Erfüllung maßgeblich zum Erfolg einer Gruppe im Rahmen des jeweiligen Problemlösungsprozesses beitragen kann. Diese Erwartungshaltung ist aber ursprünglich nicht an den Rollenträger gebunden, denn *„die erwarteten Verhaltensweisen sind die gleichen, gleichgültig, über welche persönlichen Merkmale der Rolleninhaber verfügt"* (Zimbardo & Gerrig, 2000, S.723). So schreibt auch Hare: *„The expectations shared by group members about the behaviour associated with some position in a group, no matter what individual fills the position, are called role"* (Hare, 1962, S.9). Die Erwartungen, die mit einer Rolle untrennbar verknüpft sind, werden also auf den Rollenträger übertragen, welcher die Rolle entsprechend seinen Neigungen und Fähigkeiten ausüben wird. Dies sollte bei der Besetzung von Rollen mit Gruppenmitglieder berück-

sichtigt werden: *„Es ist nun einmal so, dass bestimmte Funktionen erfüllt werden müssen, damit das Team effektiv arbeitet, und es wird immer jemand anders die Rolle übernehmen, ja nach Team, Situation, Wissensstand, Position in der Hierarchie usw.. Es ist nun Aufgabe der Gruppe [...] die richtigen Persönlichkeitstypen mit den richtigen Funktionen zu verbinden"* (Spencer & Pruss, 1995, S.58f). Auch Margerison et al. (1995) weisen auf diesen Umstand hin: *„Individuals, by nature of their work, have preferences which are a function of both their personality and work experiences, and are attracted to particular types of work"* (Margerison et al., 1995, S.15).

Es ist also offensichtlich, dass im Hinblick auf eine erfolgreiche Zielerreichung bestimmte Funktionen durch ein Team wahrgenommen werden müssen. Rollenmodelle können als eine Zusammenfassung einander ergänzender Teamfunktionen verstanden werden, wobei die tatsächlichen Ausprägungen je nach Situation und personeller Zusammensetzung variieren können, solange nur eine gewisse Ausgewogenheit herrscht und das Teamgleichgewicht hergestellt ist (vgl. Spencer & Pruss, 1995, S.92; Park & Bang, 2002, S.4; Kauffeld, 2001, S.87). Da die Wahrnehmung der notwendigen Rollen durch die Teammitglieder erfolgt, ist es nur naheliegend, dass dies möglichst in Übereinstimmung mit deren persönlichen Neigungen und Fähigkeiten erfolgen sollte. Daher enthalten Rollenmodelle wie das von Belbin (2.5.2), Margerison & McCann (2.5.3) und Spencer & Pruss (2.5.5) auch Hinweise auf typische Charaktereigenschaften geeigneter Rollenträger.

Die Definition eines jeden Rollenmodells muss natürlich stets kritisch betrachtet werden im Hinblick auf Vollständigkeit und Konsistenz der enthaltenen Rollen, insbesondere dann, wenn sich ein solches überwiegend auf empirische Beobachtungen stützt. Trotzdem stellen diese Modelle einen pragmatischen Erklärungsversuch für das Funktionieren (bzw. Scheitern) von Teams dar. Untersuchungen zu Teamzusammensetzungen und Problemsituationen liefern Hinweise zu möglichen Defiziten in der Rollenstruktur und können weiterhin als Ansatzpunkte für geeignete tutorielle Hilfestellungen dienen.

Für die Analyse von Teams, speziell im Hinblick auf die strukturelle Zusammensetzung auf der Grundlage eines Rollenmodells, stehen diverse Verfahren und Instrumente der Teamdiagnose zur Verfügung. Diese wurden im vorliegenden Kapitel charakterisiert und hinsichtlich ihrer Anwendbarkeit bewertet. Für den weiteren Verlauf dieser Arbeit sind Beobachtungen (s. Kap.2.4.3.2) und Befragungen (s. Kap.2.4.3.3) von besonderer Bedeutung. Mittels Befragungen werden beispielsweise wichtige Konzepte der VitaminL-Software überprüft und in einem iterativen Prozess bis hin zu einem *Intelligenten Tutor-System* (ITS) weiterentwickelt (s. Kap.8.2, Kap.8.3). Befragungen werden auch durchgeführt, um Hinweise auf Problemsituationen (s. Kap.11.1.3) und Hilfsstrategien (s. Kap.7.3.2) zu erhalten. Ferner wird ein spezieller Fragebogen eingesetzt, mit dessen Hilfe die Rollenprofile der Teilnehmer ermittelt werden (s. Kap.9.2.2.1, Kap.10.1.1.2). Die Grundlage bildet das Rollenmodell von Spencer & Pruss (1995) (s. Kap.2.5.5), das an die verteilte, synchrone Java-Programmierung als speziellen Kontext adaptiert wird (s. Kap.9.2). Benutzertests, die als spezielle Form der Beobachtung verstanden

werden können, werden durchgeführt, um einerseits die Praxistauglichkeit der im VitaminL-System umgesetzten Konzepte zu evaluieren, um andererseits aber auch das Verhalten der einzelnen Teilnehmer in diesem speziellen Lernkontext zu erfassen und zum Zwecke späterer Auswertungen in Protokolldateien (s. Kap.8.4.4) zu speichern. Im Rahmen der Rollenanalyse werden die auf diese Weise gewonnenen Verhaltensdaten den zugehörigen Rollenprofilen gegenübergestellt, um mittels statistischer Verfahren relevante lineare Zusammenhänge zwischen Verhaltensdaten und Rollenprofilen zu untersuchen. Verwertbare Ergebnisse fließen in die Entwicklung der Tutorkomponente (s. Kap.8.5) ein.

Kapitel 3

Computer-basierte Gruppenunterstützung

Rasante Fortschritte in den Informations- und Kommunikationstechnologien haben zu einer Durchdringung sämtlicher Bereiche von der Arbeitswelt bis zum privaten Umfeld mit eben diesen Technologien geführt. Dies wurde nicht letztlich begünstigt durch leistungsfähige Arbeitsplatzrechner und eine nahezu flächendeckende Verfügbarkeit von schnellen Internet-Zugängen bei gleichzeitiger Bezahlbarkeit seitens der Konsumenten. Das Internet bildet mit seinen Basisdiensten für den Austausch von Dateien, die Bereitstellung von Informationen und der Kommunikation zwischen Teilnehmern dabei auch die Grundlage für neue Formen sowohl des Arbeitens als auch des Lernens. Insbesondere die Unterstützung gemeinschaftlicher Zusammenarbeit[1] profitiert von einer weitgehenden Aufhebung räumlicher wie zeitlicher Grenzen: Zusammenarbeit kann jederzeit unter Einbeziehung mehrerer global verteilter Standorte stattfinden.

Die theoretischen Grundlagen dieser Form der Zusammenarbeit sind Gegenstand des interdisziplinär ausgerichteten Forschungsgebiets CSCW (*Computer Supported Cooperative Work*), welches in Kapitel 4 näher betrachtet wird. Auch das computerunterstützte Lernen in der Gruppe findet immer mehr Verbreitung. Forschungen dazu finden – ebenfalls unter Einbeziehung mehrerer voneinander unabhängiger Fachgebiete – im Gebiet des CSCL (*Computer Supported Collaborative/Cooperative Learning*) statt. Kapitel 5 widmet sich dieser Thematik.

3.1 Computer-vermittelte Kommunikation

Computer-vermittelte Kommunikation (cvK; *computer mediated communication*, CMC) wird im Rahmen dieser Arbeit verstanden als Kommunikation (s. Kap.2.2.1), die unter ausschließlicher Nutzung miteinander vernetzter Rechensysteme und darauf verfügbarer Dienste stattfindet. Dies umfasst die Kommunikation zwischen zwei oder mehr menschlichen Teilnehmern, schließt aber gleichzeitig

[1] Dieses soll, sofern nicht anders angegeben, auch das gemeinschaftliche Lernen beinhalten.

nicht-soziale Formen der Online-Kommunikation im Sinne von Mensch-Maschine-Interaktionen (wie bspw. Datenbankanfragen) aus. Wenn hier und im Folgenden von Vernetzung die Rede ist, so ist damit im Allgemeinen die Nutzung des Internet gemeint, welches derzeit de facto als *das* Netz schlechthin angesehen werden kann.

3.1.1 Das Internet und seine Dienste

Der Ursprung des Internet liegt in den 1960er Jahren im sog. ARPANET, das aus einem Projekt der Forschungsabteilung ARPA (*Advanced Research Project Agency*) des US-Verteidigungsministeriums entstand mit dem Ziel, Universitäten und Forschungseinrichtungen miteinander zu vernetzen, um somit die seinerzeit noch knappen Rechenkapazitäten sinnvoll zu nutzen. Die Integration des als Internet-Protokoll bekannten TCP/IP-Protokollstapels im Jahre 1982 führte zur Benennung als Internet. Rasanten Auftrieb erhielt das Internet ab Anfang der 1990er, als mit der Einführung des sog. WWW (*World Wide Web*) und dem ersten kostenlos verfügbaren, graphikfähigen Webbrowser das Internet für jederman bedienbar und nutzbar wurde. Die graphisch basierten Hypertextsysteme auf Basis des HTTP-Protokolls erlauben einen einfachen Zugriff auf Webseiten, die mittels sog. *Hyperlinks* untereinander verbunden sind.

Auf der Basis des Internet sind inzwischen eine Vielzahl von unterschiedlichen Diensten verfügbar, von denen nachfolgend die wichtigsten (im Kontext der vorliegenden Arbeit) skizziert werden.

Tab. 3.1: Dienste und Protokolle des Internet (Auswahl)

Dienst	**Protokoll**	**Beschreibung**
World Wide Web	HTTP,HTTPS	Übertragung von Webseiten
E-Mail	SMTP, POP3, IMAP	Versand und Empfang elektronischer Briefe (E-Mails)
Dateiübertragung *File Transfer*	FTP	Übertragung von Dateien
Namensauflösung	DNS	Umwandlung symbolischer Namen in IP-Adressen
Telnet	Telnet Protocol	Benutzung entfernter Rechner
SSH	SSH Protocol	Verschlüsselte Benutzung entfernter Rechner
Internet Relay Chat	IRC Protocol	Erster im Internet verfügbarer Chat-Dienst zum synchronen Austausch kurzer Textnachrichten
Instant Messaging	div. proprietäre Protokolle	Versenden von Kurznachrichten von Person zu Person
Internet Telefonie	SIP	Telefonieren über das Internet

Fortsetzung auf nächster Seite

Tab. 3.1: Dienste und Protokolle des Internet (Auswahl) *Forts.*

Dienst	Protokoll	Beschreibung
Usenet	NNTP	Diskussionforen zu allen erdenklichen Themen (Newsgroups)

(vgl. FUHRBERG, 2000, S.27ff)

Diese Dienste sind sowohl Grundlage als auch Ausprägungen computervermittelter Kommunikation und bilden die technische Voraussetzung für computer-unterstütztes Arbeiten und Lernen in der Gruppe. Manche Autoren differenzieren ferner zwischen genuinen Internet-Diensten, die jeweils durch ein eigenes Datenübertragungsprotokoll charakterisierbar sind, und Internet-Anwendungen, bei denen es sich um spezifische software-gestützte Anwendungsweisen von Internet-Diensten handelt.

Nachfolgende Betrachtungen werden zunächst nur auf Diensten ausgeführt, lassen sich jedoch auch auf Anwendungen, die auf diesen Diensten basieren, ausweiten.

3.1.2 Dimensionen computer-vermittelter Kommunikation

Bei der Betrachtung der verschiedenen Internet-Dienste (und -Anwendungen) zur Ermöglichung und Unterstützung computer-vermittelter Kommunikation findet üblicherweise eine Unterteilung nach mehreren Dimensionen statt. In der Regel wird hierbei nach der zeitlichen Dimension, nach der quantitativen Relation der Kommunikationsteilnehmer und nach der Darstellungsform der Kommunikationsinhalte differenziert.

3.1.2.1 Zeitliche Dimension

Die zeitliche Dimension computer-vermittelter Kommunikation entscheidet, ob die Kommunikation zwischen zwei Kommunikationspartnern zeitgleich (unter Vernachlässigung geringer, technisch bedingter Verzögerungen) oder zeitversetzt stattfindet. Dementsprechend wird zwischen synchroner und asynchroner computervermittelter Kommunikation unterschieden.

Die Verwendung asynchroner Kommunikation erfordert stets einen Kommunikationskanal, der die Nachrichten speichern kann, so dass eine gesendete Nachricht zu einem späteren Zeitpunkt empfangen werden kann. Die zeitgleiche Anwesenheit der beteiligten Kommunikationspartner ist somit nicht erforderlich. Sind die Kommunikationspartner hingegen zur selben Zeit aktiv und über einen Kommunikationskanal miteinander verbunden, so spricht man von synchroner Kommunikation.

Beide Formen sind mit Vor- und Nachteilen behaftet: *„Der Vorteil der Unmittelbarkeit, den synchrone Telekommunikation bietet, wird durch den Nachteil verringerter Zeitsouveränität erkauft"* (DÖRING, 2003, S.80), d.h. bei synchroner Kommunikation können Nachrichten nahezu zeitgleich ausgetauscht werden, so dass

sich eine Art Dialog entwickeln kann. Dies erfordert jedoch die gleichzeitige Anwesenheit aller Beteiligten.

3.1.2.2 Teilnehmerrelation

Anhand der Anzahl der Kommunikationspartner wird unterschieden zwischen Individualkommunikation (1:1), Gruppenkommunikation (n:n) und Massenkommunikation (1:N). Die Individual- oder auch interpersonale Kommunikation findet stets zwischen zwei Teilnehmern statt, an der Gruppenkommunikation sind alle Mitglieder einer beliebigen Gruppe beteiligt, wohingegen in der Massen- oder Unikommunikation Nachrichten von einem Absender zu vielen Empfängern verschickt werden.

3.1.2.3 Darstellungsform der Kommunikationsinhalte

Schließlich lässt sich eine Differenzierung bzgl. der übertragenen Inhalte bzw. ihrer Darstellungsform durchführen. Neben den klassischen textbasierten Nachrichten lassen sich auch Audiodaten (Töne, Sprache, Musik etc.), Bilddaten und Videodaten übertragen – entsprechende technische Gegebenheiten vorausgesetzt. Die Kombination mehrerer Darstellungsformen ermöglicht letztlich die Definition multimedialer Nachrichten.

3.1.3 Formen computer-vermittelter Kommunikation

Die verfügbaren Internet-Dienste und -Anwendungen lassen sich entsprechend den oben genannten Dimensionen kategorisieren (s. Tab.3.2).

Tab. 3.2: Formen computer-vermittelter Kommunikation (Auswahl)

Teilnehmer	**Asynchrone cvK**	**Synchrone cvK**
Individualkommunikation 1:1	E-Mail	Internet-Telefonie Instant Messaging
Gruppenkommunikation n:n	Mailinglisten Newsgroups	IRC-Chats Internet-Videokonferenzen Online-Spiele
Massenkommunikation 1:N	Websites	Websites

(vgl. Döring, 2003, S.125)

Dabei ist zu beachten, dass diese Einteilung nur eine grobe Kategorisierung darstellt: *„Eine trennscharfe und erschöpfende Klassifikation der einzelnen Internet-Dienste und Internet-Anwendungen ist wegen ihrer Vielfalt, Dynamik und Inte-*

grierbarkeit nicht möglich, weshalb hier mit einer heuristischen Taxonomie gearbeitet wird" (Döring, 2003, S.124). Denn wenn beispielsweise zwei Kommunikationspartner sich zeitgleich an ihren Rechnern befinden, so ist eine synchrone Kommunikation auch per E-Mail möglich.

Ein Betrachtungsschwerpunkt liegt nachfolgend auf synchroner, texbasierter Gruppenkommunikation als Bestandteil und zur Unterstützung synchroner Zusammenarbeit im Kontext der objektorientierten Programmierung gleichermaßen.

3.1.4 Strukturierte Kommunikation

Wie im Verlauf dieses Kapitels bereits aufgezeigt wurde, können die technischen Möglichkeiten, computer-vermittelt zu kommunizieren, als durchaus ausgereift betrachtet werden, sofern man die von Zeit zu Zeit auftretenden störenden Nebeneffekte außer Acht lässt, die sich aus der unvermeidbaren Reduktion der Kommunikationskanäle ergeben (vgl. Döring, 2003, Kap.3.2.1). Trotz dieser Möglichkeiten existieren Situtationen respektive Anwendungsszenarien, die eine künstliche Beschränkung des technisch Machbaren rechtfertigen.

In einigen speziellen Computer-Anwendungen sind Varianten von textbasiertem Chat umgesetzt, bei denen eine feste Auswahl an Kommunikationselementen vorgegeben ist. Dies hat zur Folge, dass sämtliche Kommunikation zwischen den Kommunikationspartnern ausschließlich auf der Basis einer fest-definierten endlichen Menge von *conversational elements* (vgl. Soller, 2004, S.358), sprich strukturiert erfolgen kann. Diese spezielle Form der computer-vermittelten Kommunikation wird daher nachfolgend auch als *strukturierte Kommunikation* bezeichnet.

Die technische Realisierung erfolgt üblicherweise, indem geeignete Objekte einer Benutzungsschnittstelle (wie bspw. Schaltflächen/Buttons, Menüpunkte, Auswahllisten/Comboboxes) auf die jeweiligen Kommunikationselemente abgebildet werden. Nach Auswahl beziehungsweise Betätigung eines Interaktionsobjekts durch den Anwender wird eine dem gewählten Kommunikationselement entsprechende Botschaft generiert und an den oder die Kommunikationspartner versendet, wobei – abhängig vom zugrundeliegenden Modell strukturierter Kommunikation und vom aktuell gewählten Element – der Anwender die eigentliche Botschaft vor dem Versand um eigene Textinhalte ergänzen kann.

Die Umsetzung eines Modells strukturierter Kommunikation in Form einer Benutzungsschnittstelle wird *strukturierte Kommunikationsschnittstelle* oder auch *strukturierte Dialogschnittstelle* genannt. Ein Beispiel einer solchen strukturierten Kommunikationsschnittstelle ist in der Abbildung 3.1 dargestellt. Der in der vorliegenden Arbeit verwendete Ansatz strukturierter Kommunikation, die *Collaborative Learning Skills* McManus & Aiken (1995), wird nun näher vorgestellt, weitere Beispiele strukturierter Kommunikationsschnittstellen werden im Zusammenhang mit ihren Anwendungen im Kapitel 6.4 kurz skizziert.

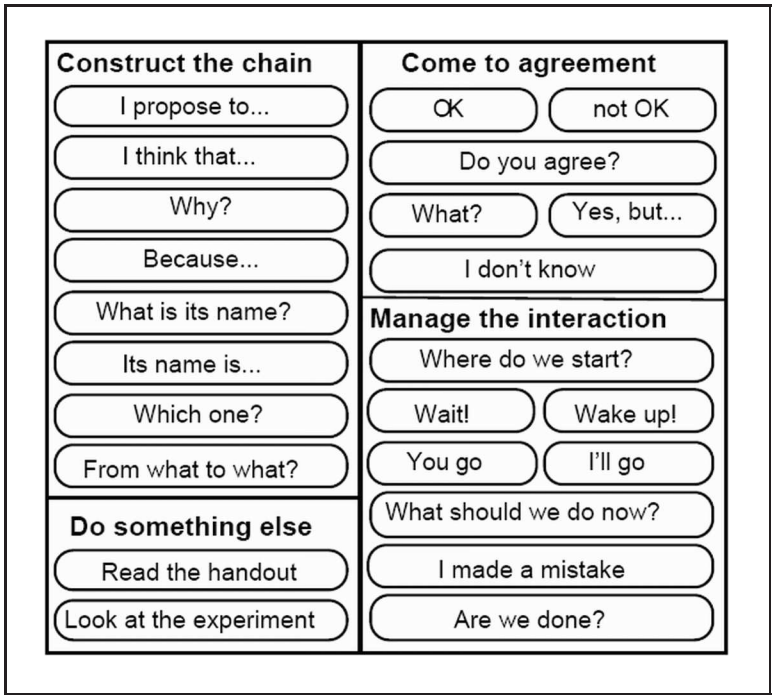

Abb. 3.1: Beispiel einer strukturierten Kommunikationsschnittstelle (Baker & Lund, 1997, S.183)

3.1.4.1 Collaborative Learning Skills

Funktionierende Gruppenprozesse – und damit im Wesentlichen auch effektive Kommunikation – bilden die unverzichtbare Grundlage erfolgreicher kollaborativer Lernprozesse. Diverse Autoren räumen den Erklärungen dabei einen besonderen Stellenwert ein: Als Antworten auf Hilfegesuche und als Ergänzungen zu Hilfestellungen anderer Gruppenmitglieder sind Erklärungen ein wesentlicher und unverzichtbarer Bestandteil eines interaktiven Lernprozesses, an dessen (erfolgreichen) Ende innerhalb der Lerngruppe etwaige Meinungsverschiedenheiten beseitigt sind und somit Konvergenz hinsichtlich einer Problemlösung in Form einer gemeinsamen Wissensbasis erzielt wird (vgl. McManus & Aiken, 1995, S.309).

Diese diskursiven Lernprozesse sind von Johnson & Johnson (1991) eingehend analysiert worden. Sie konnten dabei feststellen, dass *„in their explanations and discussions, students use collaborative skills, characterized by appropriate phrases or sentence openers. For example, the sentence opener «I think» indicates the openness attribute in communication skills, and the sentence opener «But what I don't understand is how ...» indicates a request for an explanation"* (McManus & Aiken, 1995, S.309). Es lassen sich schließlich Kommunikationselemente identifizieren, die für die erwähnten Lernprozesse charakteristisch sind. Johnson & Johnson (1991) konstruieren aus diesen eine Taxonomie von für diese Form der Zusammenarbeit typischen Gesprächselementen, den sogenannten *Collaborative Skills*. Diese Einteilung ist hierarchisch strukturiert und besteht auf der obersten Ebene aus den Kompetenzen *Leadership* (Führungsstil), *Trust* (Vertrauen),

Communication (Kommunikation) und *Creative Conflict* (Kreativer Konflikt). Jede dieser Hauptkompetenzen (*Skills*) setzt sich wieder zusammen aus mehreren *Subskills*, welche ihrerseits aus diversen Attributen (*attributes*) bestehen. Diesen sich auf der untersten Ebene der Taxonomie befindlichen Attributen wiederum ordnen die Autoren geeignete Satzanfänge zu. Insgesamt wird somit nicht nur eine begrenzte Menge von Konversationselementen mit zugehörigen Phrasen definiert, sondern auch eine einfache Codierung von Diskussionsbeiträgen ermöglicht.

Im Rahmen ihrer eigenen Forschungen im Bereich computer-gestützten kollaborativen Lernens (vgl. Kap.5) verwenden McManus & Aiken (1995) den Ansatz von Johnson & Johnson (1991), um Studierende bei computer-basierten Problemlösungsprozessen zunächst zu beobachten und nachfolgend zu unterstützen. Dabei bearbeiten die Teilnehmer paarweise, in wechselnden Gruppen Teilprobleme eines Projektes, an dessen Ende eine Statistikanwendung steht, die neben mehreren Algorithmen (wie bspw. Mittelwertberechnung oder Standardabweichung) auch die Ausgabe der berechneten Ergebnisse beinhaltet. Die einzelnen Arbeitsphasen werden unter Zuhilfenahme einer besonderen Anwendung, eines *Intelligent Collaborative Learning System* (ICLS; vgl. McManus & Aiken, 1995, S.307), durchgeführt, die die Kommunikation – und damit auch die Zusammenarbeit – mittels einer speziellen Software-Komponente unterstützt: Der *Group Leader Tutor* stellt in diversen Phasen und Problemsituationen während der Zusammenarbeit geeignete Unterstützungsfunktionen bereit (vgl. McManus & Aiken, 1995, S.313), die in Form von Skripten beschrieben sind und mit sogenannten *Deterministischen Endlichen Automaten* (DEA) umgesetzt werden.

Die prinzipielle Arbeitsweise eines solchen Skripts (bzw. des zugehörigen Automaten) lässt sich wie folgt skizzieren: Die Diskussionsbeiträge der Lernenden während einer Sitzung werden mit einem Parser syntaktisch analysiert, auf die Taxonomie der *Collaborative Skills* abgebildet und somit codiert. Der resultierende Code wird vom Endlichen Automaten als Eingabe verarbeitet. Dort erfolgt anhand dieser Eingabe eine Transition aus einem aktuellen Zustand in einen Folgezustand: *„In the finite state machine, each circle represents a discussion state and each arc represents a sentence opener, indicating a collaborative skill, used as input. The machine changes from the current state to the next state based on the sentence opener used. States may be intermediate states, final states, or error states, depending on the sentence opener chosen as the discussion progresses to convergence"* (McManus & Aiken, 1995, S.318). Die fehlerhafte Verwendung von Satzanfängen führt folglich zu einem Fehlerzustand, in welchem der *Group Leader Tutor* korrigierend eingreift. Ausführliche Informationen hinsichtlich Definitionen, Arbeitsweisen und Einsatzgebieten von DEAen sind u.a. in Hopcroft et al. (2002) und Aho et al. (1988a,b) zu finden.

Trotz einiger Beschränkungen durch die seinerzeitigen technischen Begebenheiten (geringe Ausführungsgeschwindigkeiten, Begrenzung auf zwei Teilnehmer pro Gruppe, text-basierte Benutzungsoberfläche ohne graphische Elemente) zeigt sich das System von McManus & Aiken (1995) als geeignet bei der Förderung von kommunikativen Fähigkeiten, die sich für kollaborative Prozesse – wie die gemein-

same Problemlösung – hilfreich erweisen: *„Though this experiment was conducted with a small number of subjects, it provided preliminary evidence that the ICLS can be useful in assisting students to develop collaborative skills. Therefore, this system could be useful both for students working locally in groups and for those working in virtual classrooms through distance education"* (McManus & Aiken, 1995, S.318).

Der soeben skizzierte Ansatz strukturierter Kommunikation als eine spezielle Form computer-vermittelter Kommunikation in Form von *Collaborative Skills* und entsprechenden Satzanfängen wird in einem weiteren ICLS im Rahmen des Projekts *EPSILON* (s. Kap.6.4.4) aufgegriffen und für die Domäne der objektorientierten Modellierung adaptiert. Auf weitere Systeme, die verschiedene Ansätze strukturierter Kommunikation verwenden, wird in Kapitel 6.4 näher eingegangen.

3.1.4.2 Nachteile strukturierter Kommunikation

Die Verwendung strukturierter Kommunikation ist nicht ganz unkritisch zu betrachten, findet doch zweifelsohne eine weitere Beschränkung der Ausdrucksmöglichkeiten statt, da die teilnehmenden Kommunikationspartner ihre Beiträge unter ausschließlicher Zuhilfenahme vorgegebener Formulierungen verfassen können. Dabei handelt es sich um ganze Sätze oder um Satzanfänge, die vom Benutzer vervollständigt werden können. Dies führt gelegentlich dazu, dass Benutzer unpassende Elemente auswählen und damit den Ansatz der strukturierten Kommunikation sinnverfälschend einsetzen: *„The system codes were obtained directly from the sentence openers that students choose to begin their contributions, and may not accurately reflect the intention of the contribution. For example, a student might choose the opener, «I think», and then add, «I disagree with you». Each sentence opener is associated with one subskill and attribute pair that most closely matches the expected use of the phrase; however even having gone through sentence opener training ..., students may not always use the openers as expected"* (Soller et al., 2002, S.7). Aus solch einer Verwendung der gegebenen Kommunikationselemente ergeben sich entsprechend falsche Codes, so dass ein verarbeitendes System (wie bspw. der *Group Leader tutor* von McManus & Aiken (1995)) nicht angemessen reagieren kann, schlimmstenfalls sogar kontra-produktiv agiert und den Teilnehmern fehlerhafte Unterstützung liefert.

Es muss an dieser Stelle ganz klar betont werden, dass die mit der Verwendung von strukturierter Kommunikation einhergehende Restriktion im Rahmen der vorliegenden Arbeit geduldet wird, da es sich hierbei um einen ersten Ansatz handelt, der lediglich ein Werkzeug zur Analyse der Kommunikation darstellt. Es kann Verwendungen geben, die diesem Werkzeug nicht 100%ig entsprechen. Aus diesem Grunde wird eine Einarbeitungsphase als Vorbereitung für einen sinnvollen Einsatz solch einer strukturierten Kommunikationsschnittstelle als notwendig erachtet: *„Although some of the subjects found having to choose a sentence opener somewhat restrictive, most subjects became more comfortable with the interface once they had a chance to experiment with it"* (Soller, 2001, S.13). Vorkenntnisse

im Umgang mit herkömmlichen (sprich unstrukturierten, nicht-restriktiven) Chat-Anwendungen werden hierbei als hilfreich erachtet (vgl. Soller et al., 1999, S.6).

3.1.4.3 Vorteile strukturierter Kommunikation

Trotz der genannten Probleme besitzen die diversen Ansätze strukturierter Kommunikation (vgl. Baker & Lund, 1997; Jermann, 1999; Baker et al., 2001; Matessa, 2001; Soller & Busetta, 2003) auch positive Potenziale, die die aufgeführte negative Kritik kompensieren und eine Verwendung rechtfertigen. Insbesondere wird durch die Vorgabe einer endlichen Menge von konversationalen Elementen die Möglichkeit geschaffen, eine Codierung durchzuführen, d.h. jeder Diskussionsbeitrag eines Kommunikationsteilnehmers wird direkt abgebildet auf ein Codewort aus dem zugehörigen Code. Dies resultiert letztlich in einer starken Vereinfachung der Analyse von Diskussionsverläufen durch ein entsprechendes Software-System. So stellt auch Soller (2004) in ihren Arbeiten, die auf der Verwendung des Ansatzes der *Collaborative Learning Skills* nach McManus & Aiken (1995) beruhen, fest: *„In this research, requiring students to use a given set of sentence openers allowed the system to automatically code the dialog without having to rely on Natural Language parsers"* (Soller, 2004, S.355). Als weitere Folge beim Einsatz strukturierter Kommunikation lassen sich die Intentionen der Teammitglieder derzeit gezielter aus den Diskussionsbeiträgen der Teilnehmer ableiten (vgl. Matessa, 2001, S.3). Auch Probleme, die während der Zusammenarbeit auftreten, sind besser identifizierbar (vgl. Soller, 2001, S.8).

Bei der Verwendung strukturierter Kommunikation konnten sowohl Soller (2001) als auch Baker et al. (2001) – trotz unterschiedlicher Ansätze (s. Kap.6.4.4 und Kap.6.4.3) – feststellen, dass die Diskussionsbeiträge derjenigen Teilnehmer, die eine strukturierte Dialogschnittstelle verwenden, stärker auf die eigentliche Aufgabe fokussiert sind, wohingegen eine traditionelle Chat-Kommunikation einen vergleichsweise höheren Anteil an sogenannten *off-topic*-Beiträgen[2] aufweist: *„This second dedicated button interface was also experimented with four dyads for the same task. Results of comparison with the chat-box interaction corpus were revealing and encouraging. Despite the fact that the average number of acts devoted to managing the interaction remained largely unchanged, with the dedicated interface the students engaged in a more task-focussed interaction"* (Baker et al., 2001, S.2). Im Sinne einer möglichst erfolgreichen Aufgabenbearbeitung sind solche Beiträge verständlicherweise unerwünscht.

Letztlich kann dieser Ansatz nicht nur für die Kommunikation zwischen menschlichen Teilnehmern verwendet werden, sondern darüber hinaus auch als Grundlage der Kommunikation von künstlichen Teammitgliedern (insbesondere virtuellen Tutoren; s. Kap.6) mit den menschlichen Kommunikationspartnern dienen. Es ergibt

[2] Der Begriff *off topic* entstammt der englischen Sprache und bedeutet sinngemäß *am Thema vorbei* oder *ohne Bezug zum Thema*. Speziell bei der cvK werden damit Email- oder Chat-Beiträge bezeichnet, die sich nicht mit dem eigentlichen Thema befassen.

sich eine Reduktion der möglichen Unterschiede zwischen den Teilnehmern, was insbesondere dann hilfreich sein kann, wenn (menschliche) Teilnehmer Vorbehalte gegenüber künstlichen Tutoren besitzen. So stellte MATESSA (2001) in seinen Forschungen fest: *„In addition, roughly half of the subjects interacting with intelligent agents thought they were working with human subjects (there is still room for improvement, however, since only 10 % of subjects interacting with human partners thought their partners were computers)"* (MATESSA, 2001, S.3).

Da in diesem Kontext das Parsing von natürlich-sprachlichen Eingaben nicht im Fokus steht, wird als Behelfsmaßnahme das Konzept einer strukturierten Kommunikation weiterverfolgt und für die in dieser Arbeit eingesetzten VitaminL-Applikation (s. Kap.8) entsprechend umgesetzt.

3.2 Virtuelle Gruppen

Die im vorigen Kapitel genannten Dienste ermöglichen die Bildung von Gruppen, deren Mitglieder nicht mehr notwendigerweise in direkten Kontakt miteinander treten, ja sich noch nicht einmal mehr persönlich kennen müssen. Es kann eine räumliche – und je nach verwendetem Dienst möglicherweise auch zeitliche – Entkopplung des Einzelnen von seiner Gruppe stattfinden. Damit ergeben sich neue Gruppen, die die bestehenden Gruppenformen (s. Kap.2.1.2) um sog. *virtuelle Gruppen* erweitern: *„Soziale Gruppen, deren Mitglieder sich im Netz erstmals begegnen und deren wechselseitige Kommunikation überwiegend netzbasiert abläuft, bezeichnet man als »virtuelle Gruppen«"* (DÖRING, 2003, S.501).

Solche Gruppen werden deshalb als virtuell bezeichnet, weil die sozialen Interaktionen zwischen den Gruppenmitgliedern nicht unmittelbar, sondern mittels digitaler Repräsentationen stattfinden. Dabei hat der Zusatz *virtuell*, der seine Herkunft im lateinischen Wort *virtus* (dt.: Stärke, Kraft, Tugend) bzw. im mittelenglischen Begriff *virtual* (dt.: »dem Vermögen nach«, »der Möglichkeit nach vorhanden, aber nicht aktuell wirksam« oder auch »eigentlich«) besitzt, längst seine ursprüngliche Bedeutung: *„Virtuell ist alles, nicht nur das isomorphe Modell des Realen, sondern auch der Gang durch die künstliche Abbildung, als virtuell gelten nicht nur die in den elektronischen Raum transferierten Institutionen, die Online-Seminare und virtuellen Universitäten, sondern auch die Nutzung derselben zum Zwecke des Lernens"* (SCHULMEISTER, 2001, S.221).

Die Virtualisierung stellt demnach einen Übergang von der Realität in die Virtualität dar. Dabei wird aus dem ursprünglich unmittelbaren Wahrnehmen und Agieren zwischen Gruppenmitgliedern innerhalb einer realen Umgebung ein mittelbares Wahrnehmen und Agieren über entsprechende Ein-/Ausgabegeräte (s. Abb.3.2).

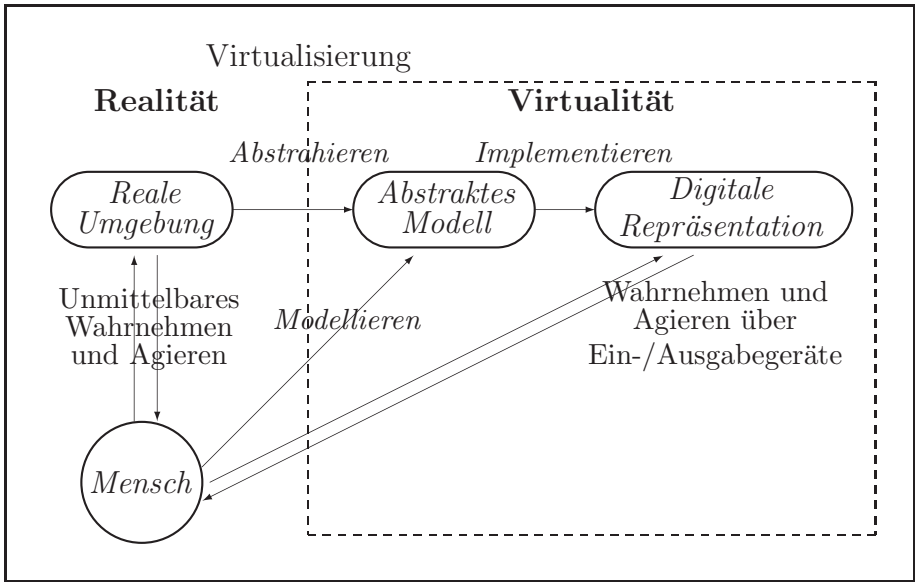

Abb. 3.2: Der Prozess der Virtualisierung
(vgl. Konradt & Hertel, 2002, S.13)

Virtuelle Organisationsformen lassen sich anhand der Gruppengröße in *virtuelle Teams* und *virtuelle Gemeinschaften* unterscheiden, wobei sich letztere in Anlehnung an die Taxonomie nach (Döring, 2003, S.490ff) weiter untergliedern lassen. Unabdingbar für die Konstitutionierung einer virtuellen Organisationsform ist ein *virtueller Treffpunkt*, also ein technisches System, das über eine den Mitgliedern bekannte (Internet-)Adresse verfügt und entsprechende synchrone und asynchrone Internet-Dienste anbietet. Der Umfang der verfügbaren Dienste kann sich von einfachen Internet-Anwendungen (z.B. Mailinglisten) bis zu sog. *Community-Plattformen* mit komplexen Funktionen für Kommunikation, Kollaboration und Mitgliederverwaltung erstrecken.

3.2.1 Virtuelle Gemeinschaften

Virtuelle Gemeinschaften (*virtual/online/cyber/electronic communities*) sind als Ergebnis der Gruppenbildung mittels computer-vermittelter Kommunikation solche Gemeinschaften, deren Mitglieder sich ausschließlich oder überwiegend über ein Netzwerk (in der Regel das Internet) treffen. Die Regelmäßigkeit der Teilnahme sowie die Qualität der Beiträge entscheidet dabei oft über den Bindungsgrad zwischen dem einzelnen Mitglied und der jeweiligen Community: *„Viele virtuelle Orte sind wie Parks, Schwimmbäder oder Kneipen für alle Internet-Nutzer/innen öffentlich zugänglich, so dass sie ungehindert das Geschehen beobachten und sich auch einmischen können. Damit ist man jedoch weder im eigenen Empfinden noch für die anderen Beteiligten bereits ein »Mitglied« der jeweiligen virtuellen Gruppe. Vielmehr verläuft der Weg vom Passanten oder »Touristen« zum eingesessenen Gruppenmitglied über regelmäßiges Engagement, kompetente Beiträge und das Knüpfen von Kontakten und Beziehungen"* (Döring, 2003, S.502). Für

die überwiegende Zahl interessierter und aktiver Mitglieder besitzt eine virtuelle Gemeinschaft daher den Charakter einer Primärgruppe (s. Kap.2.1.2.3).

Trotz der Notwendigkeit eines geeigneten technischen Systems zur Realisierung des virtuellen Treffpunkts ist letztlich der thematische Fokus ausschlaggebend für die Teilnahme an einer solchen Gemeinschaft, die sich durch die dargebotenen Inhalte auch von anderen Gemeinschaften abgrenzt. Nach diesem Fokus kann man virtuelle Gemeinschaften weiterhin unterscheiden in formale und informelle virtuelle Gemeinschaften, also beispielsweise in *Business Communities* und *Fun Communities*.

3.2.2 Virtuelle Teams

„Man spricht von »virtuellen Teams«, wenn eine überschaubare Zahl von Personen computervermittelt zusammenarbeitet, um gemäß einem festen Zeitplan gemeinsam eine umschriebene Aufgabe zu erledigen" (Döring, 2003, S.520). Eine ähnliche Definition findet sich in Konradt & Hertel (2002): *„Als virtuelle Teams werden flexible Gruppen standortverteilter und ortsunabhängiger Mitarbeiterinnen und Mitarbeiter bezeichnet, die auf der Grundlage von gemeinsamen Zielen bzw. Arbeitsaufträgen ergebnisorientiert geschaffen werden und informationstechnisch vernetzt sind"* (Konradt & Hertel, 2002, S.18). Im Unterschied zu den eben genannten virtuellen Gemeinschaften handelt es sich bei einem virtuellen Team folglich um eine Kleingruppe, üblicherweise um eine formale Sekundärgruppe, da die Gruppenbildung aufgrund eines äußeren Anlasses erfolgt.

Ganz klar im Vordergrund steht bei virtuellen Teams also weniger die Wahrnehmung und Pflege individiueller Interessen und Vorlieben, als vielmehr die gemeinschaftliche Aufgabenerfüllung, die sowohl in einen betriebswirtschaftlichen Kontext eingebettet sein kann, sich aber auch auf eine Lernsituation übertragen lässt. Wenn dies nun unter vorwiegender Nutzung von Informations- und Kommunikationstechnologien geschieht, wird diese Form der Zusammenarbeit auch virtuell bzw. *telekooperativ* genannt: *„Telekooperation bezeichnet eine mediengestützte arbeitsteilige Leistungserstellung von individuellen Aufgabenträgern, Organisationseinheiten und ganzen Organisationen, die über mehrere Standorte verteilt sind"* (Konradt & Hertel, 2002, S.13). Die Existenz virtueller Teams ist nicht ausschließlich auf ein betriebliches Umfeld beschränkt, auch das Lernen in der Gruppe kann in virtueller Form stattfinden.

3.2.2.1 Virtuelle Arbeitsgruppen

Virtuelle Teams (im Sinne virtueller Arbeitsgruppen) ermöglichen ihren Mitgliedern eine räumliche Entkopplung von der Gruppe und damit auch eine Loslösung der Arbeit des Einzelnen aus dem Gesamtprozess, so dass die Leistungserbringung nicht mehr an die betriebliche Arbeitsstätte gebunden ist. Die organisatorische Grundlage aller telekooperativen Arbeitsformen ist die Telearbeit: *„Unter Telearbeit werden Erwerbstätigkeiten verstanden, die zumindest teilweise außerhalb der*

bisherigen Betriebsstätten durch elektronische Informationstechnologien und deren Dienste verrichtet werden" (Konradt & Hertel, 2002, S.15).

Diese Form virtueller Arbeit lässt sich nicht nur auf die ganze Gruppe anwenden, vielmehr ist eine Ausweitung der virtuellen Kooperation auch über Gruppengrenzen hinaus möglich. Prinzipiell kann virtuelle Kooperation sowohl innerhalb einer Gesamtorganisation als auch zwischen Organisationen stattfinden. Konradt & Hertel (2002) definieren ein *virtuelles Unternehmen* als *„eine Kooperationsform rechtlich unabhängiger Unternehmen, Institutionen und/oder Einzelpersonen, die eine Leistung auf der Basis eines gemeinschaftlichen Geschäftsverständnisses erbringen. Die kooperierenden Einheiten beteiligen sich an der Zusammenarbeit vorrangig mit ihren Kernkompetenzen und wirken bei der Leistungserstellung gegenüber Dritten wie ein einheitliches Unternehmen. Dabei wird auf die Institutionalisierung zentraler Managementfunktionen zur Gestaltung, Lenkung und Entwicklung des virtuellen Unternehmens durch die Nutzung geeigneter Informations- und Kommunikationstechnologien weitgehend verzichtet"* (Konradt & Hertel, 2002, S.20).

Diese und weitere Formen der Telearbeit stehen offensichtlich in hierarchischen Beziehungen zueinander: So ist die Telearbeit des Einzelnen Grundlage für die Konstitutionalisierung virtueller Teams, welche ihrerseits als Voraussetzung für die Bildung virtueller Unternehmen anzusehen sind. Eine Einteilung nach den Dimensionen *Ausmaß der Kooperation* und *Integration der Telekooperation* ist der Abbildung 3.3 zu entnehmen (vgl. Konradt & Hertel, 2002, S.23).

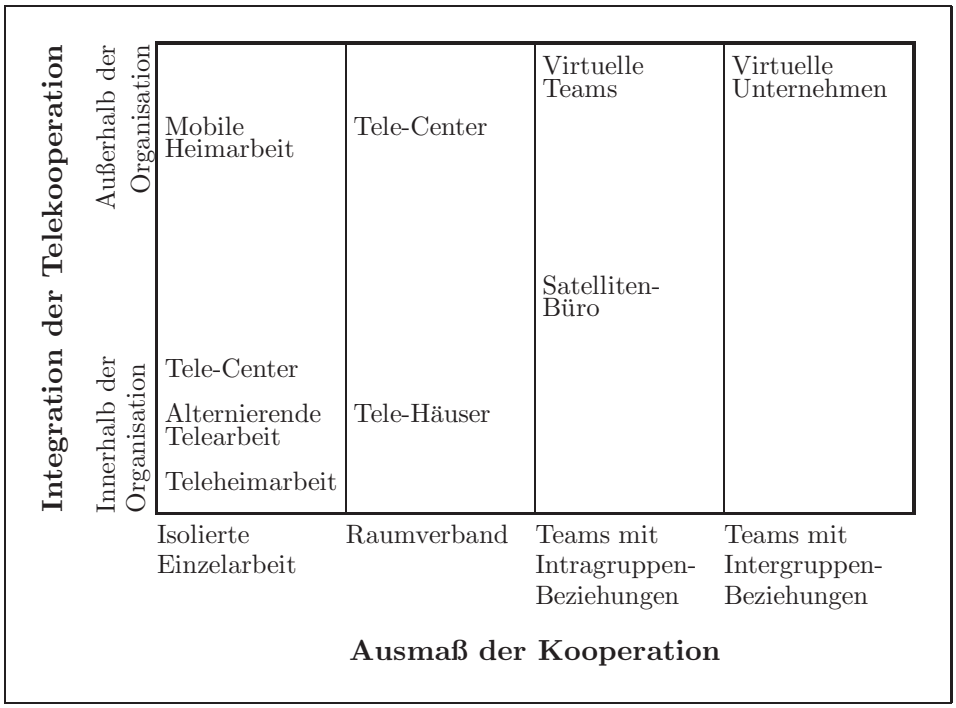

Abb. 3.3: Virtuelle Arbeitsformen
(vgl. Konradt & Hertel, 2002, S.23)

Die virtuellen Teams, denen als Betrachtungsgegenstand der vorliegenden Arbeit

besonderes Augenmerk gebührt, sind also gekennzeichnet durch einen sehr hohen Integrationsgrad computer-unterstützter Kooperation, d.h. die Zusammenarbeit innerhalb eines virtuellen Teams findet zu wesentlichen Teilen computervermittelt statt. Dabei kommen die in Kapitel 3.1 genannten Dienste zum Einsatz, die in der Regel miteinander kombiniert werden. Es entstehen dabei mehr oder weniger komplexe, leistungsfähige Software-Systeme, die unter dem Begriff *Groupware* zusammengefasst werden (s. Kap.4.5). Den theoretischen Rahmen bildet das Forschungsgebiet CSCW (s. Kap.4).

3.2.2.2 Virtuelle Lerngruppen

Virtuelle Lerngruppen als Sonderfall virtueller Teams sind folglich Lerngruppen (s. Kap.2.1.4), deren Lernsituation vollständig außerhalb der Gruppe selber stattfindet, d.h. der Wissensaustausch zwischen den Gruppenmitgliedern findet komplett unter Zuhilfenahme von computer-vermittelter Kommunikation und deren Diensten statt (s. Kap.3.1; vgl. Döring, 2003, S.520f).

Die Disziplin, die sowohl die technischen Systeme wie auch die pädagogisch-didaktischen Aspekte behandelt, wird mit dem Akronym CSCL benannt (s. Kap.5). Ihre Inhalte umfassen Fragen zur Förderung von (virtuellen) Lerngruppen unter Einsatz von Computersystemen in Kombination mit Methoden aus der Pädagogik und der Didaktik (vgl. Pfister & Wessner, 2001b, S.251).

3.2.3 Konsequenzen

Die Nutzung von Computern und Internet-Diensten hat nicht nur die Virtualisierung von Gruppenarbeit (inkl. Gruppenlernen) ermöglicht, sondern sowohl herkömmliche wie auch virtuelle Arbeitsformen beeinflusst. Neben potentiellen Vorteilen, die sich daraus ergeben können, dürfen auch mögliche Nachteile nicht unerwähnt bleiben. Des weiteren ist zu beachten, dass mit dieser speziellen Arbeitsform auch besondere Anforderungen an die virtuellen Organisationsformen und deren Mitglieder verbunden sind.

3.2.3.1 Vorteile virtueller Gruppenarbeit

Ein wesentlicher Vorteil bei der Nutzung computer-vermittelter Kommunikation über das Internet besteht in der schnellen und kostengünstigen Übermittlung von Daten, Nachrichten und Informationen, d.h. der globale Versand von Informationen kann innerhalb kürzester Zeit durchgeführt werden. Bei konsequenter Nutzung digitaler Daten lassen sich Medienbrüche (in Form von Ausdrucken auf Papier) vermeiden und es besteht überdies die Möglichkeit, die digitalen Daten zahlreichen weiteren Anwendungen zuzuführen oder aber auch, diese als Grundlage für Kooperationen mit Lieferanten und/oder Kunden einzusetzen (vgl. Konradt & Hertel, 2002, S.24f). *„Diese Vorteile sind auch dann nutzbar, wenn keine telekooperative Arbeit im engeren Sinne verrichtet wird. Sekretariats- oder Fachaufgaben*

können durch das Internet beschleunigt und Workflows effizienter gestaltet werden, ohne die Arbeits- oder Organisationsform selbst zu ändern" (Konradt & Hertel, 2002, S.25).

Konzentriert man seine Betrachtung auf rein virtuelle Arbeits- und Organisationsformen, so können weitere positive Aspekte beobachtet werden. So kann sich als Konsequenz aus der bereits genannten räumlichen Entkopplung (s. Kap.3.2.2.1) der Wegfall von Anfahrtswegen ergeben, verbunden mit einer Reduzierung von Fahrtkosten. Weitere Kostensenkungen sind aufgrund geringerer Lohnnebenkosten und von Einsparungen bei Raumkosten am Unternehmensstandort denkbar.

Neben der bereits erwähnten Standortunabhängigkeit wird als weiterer Vorteil oft die Flexibilisierung von Arbeitszeiten genannt, die es den Mitgliedern virtueller Teams gestattet, die Zeiträume zur Erbringung ihrer Individualleistungen an der Gesamtarbeit eigenverantwortlich – unter Berücksichtigung gruppenspezifischer Rahmenbedingungen wie Abgabetermine und Gruppensitzungen – zu gestalten. Als Folge größerer Zeitsouveränität kann häufig eine gestiegene Arbeitsmotivation der Mitarbeiter festgestellt werden (vgl. Konradt & Hertel, 2002, S.31).

Die konsequente Nutzung computer-vermittelter Kommunikation ermöglicht ferner eine maximale Informationsversorung aller Beteiligten sowie eine schnellere und direkte Informationsweitergabe. Als Folge davon kann oftmals der Wegfall von Hierarchien beobachtet werden.

Letztlich sind weit größere Freiheiten bezüglich der Teamzusammensetzung feststellbar: Bei der Wahl von Mitarbeitern spielen Entfernungen nur noch eine untergeordnete Rolle, so dass – zeitliche Verfügbarkeit vorausgesetzt – virtuelle Teams aus den fachlich besten Mitarbeitern gebildet werden können, selbst wenn diese auf verschiedene Standorte verteilt sind. Auch die Integration freier Mitarbeiter wird erleichtert.

3.2.3.2 Probleme virtueller Gruppenarbeit

Die Virtualisierung kann neben den genannten Vorteilen auch einige Nachteile mit sich bringen, die bei der Planung und Durchführung virtueller Gruppenarbeit Berücksichtigung finden müssen, um den Erfolg weiterhin zu gewährleisten. Zunächst einmal erfolgt aus dem konsequenten Einsatz von Informations- und Kommunikationstechnologien eine weitgehende Abhängigkeit von diesen. Daher sollte im Bedarfsfall auf alternative Kommunikationswege zugegriffen werden können, um somit durch Redundanz die Abhängigkeit zu reduzieren. Da die Zusammenarbeit computervermittelt erfolgt, ergeben sich dabei weniger direkte Kontakte mit Kollegen: Es entfallen beispielsweise informelle Treffen auf dem Gang oder am Kaffeeautomaten, was schlimmstenfalls zur sozialen Isolation und der Abkopplung Einzelner von betrieblichen Entwicklungen nach sich zieht. Aus dieser Situation kann sich eine verringerte Identifikation mit dem Unternehmen ergeben (vgl. Konradt & Hertel, 2002, S.33).

Im Falle einer Verlagerung der Arbeit in das häusliche Umfeld (wie dies bspw.

bei der Telearbeit geschieht) sind unter Umständen negative Auswirkungen sowohl auf die Arbeitserfüllung wie auf die familiäre Situation feststellbar, da die ansonsten gegebene Trennung zwischen Arbeitsplatz und Wohnung aufgehoben wird, was sich negativ auf Leistung, Leistungsbereitschaft und das Befinden niederschlagen kann (vgl. KONRADT & HERTEL, 2002, S.29). Auch die Möglichkeiten der Kontrolle (bspw. durch den Arbeitgeber) sind nur eingeschränkt vorhanden, bedingt u.a. durch rechtliche Rahmenbedingungen wie der Unverletztlichkeit der Wohnung (Artikel 13 Grundgesetz; vgl. KONRADT & HERTEL, 2002, S.34). Als Folge mangelnder Kontrollmöglichkeiten kann Missbrauch bzgl. Zeitabrechnung und -verwendung auftreten. Auch ist der Grad der Zielerreichung bzw. Aufgabenerfüllung schlechter kontrollierbar. Hier sind entsprechende Maßnahmen im Vorfeld zu treffen wie beispielsweise der Einsatz geeigneter Führungsstile: *„Dabei muss berücksichtigt werden, dass die Notwendigkeit zur Anwesenheits- und Zeitkontrolle nur bei Führungsstilen gegeben ist, die eine Anwesenheit erfordern. Bei delegativen Führungskonzepten wie Management by objectives werden die Zielabsprache und die Terminierung und Feinsteuerung weitgehend selbst überlassen. Die Kontrolle erfolgt in Form von regelmäßigen Gesprächen"* (KONRADT & HERTEL, 2002, S.34).

Treffen aller Teammitglieder virtueller Teams sind – im Vergleich zu traditionellen Teams, deren Mitglieder an einem Standort versammelt sind – mit einem erhöhten Aufwand verbunden. Dieser kann die Organisation betreffen, die zur Planung und Durchführung von Treffen notwendig ist, aber auch die technischen Maßnahmen berühren, die beispielsweise im Falle von Videokonferenzen durch Anschaffung und Bereitstellung entsprechender Geräte an allen teilnehmenden Standorten durchzuführen sind.

Auch die Bildung virtueller Teams ist mit einem erhöhten organisatorischen Aufwand behaftet: *„Um virtuelle Teams zu initialisieren ist es wichtig, dass die Teammitglieder sich einander vorstellen, dass die Zielsetzung transparent ist und auch die zentralen Arbeitsweisen erklärt werden – dies betrifft etwa Medienwahlen, Zeitstrukturen, Rollenverteilungen usw. Werden spezielle Systeme zur Unterstützung eingesetzt, so ist in der Forming-Phase der Gruppe auch die Aneignung dieser Konferenz- oder Kollaborations-Technik vordringlich. Wenn kein Face-to-Face-Treffen möglich ist, sollten doch auf jeden Fall synchrone Online-Meetings die Gruppenmitglieder regelmäßig zusammenbringen"* (DÖRING, 2003, S.522). Wie in herkömmlichen Gruppen auch, sind Maßnahmen zur Bildung und Aufrechterhaltung von Vertrauen und Kooperation unverzichtbar, gleichwohl sie in virtuellen Teams schwieriger durchzuführen sind (vgl. KONRADT & HERTEL, 2002, S.33).

Dieser erhöhte Aufwand bzgl. Organisation und Kommunikation betrifft auch Lerngruppen: *„Die Initialisierung und Organisation virtueller Lerngruppen erfordert in der Regel besonderen Einsatz von Seiten der Dozentin bzw. des Dozenten: Man muss sich nicht nur um technische Belange und Probleme der Teilnehmer kümmern, sondern auch dafür sorgen, dass die netzvermittelten Kommunikationsprozesse in Gang kommen, und sieht sich somit gezwungen, selbst sehr häufig Beiträge zu schreiben (z.B. damit Anfragen nicht unbeantwortet bleiben oder keine*

überlangen Kommunikationspausen enstehen" (Döring, 2003, S.521).

Auch die Vermeidung und Behandlung von Konflikten muss im Falle virtueller Teams berücksichtigt werden: *„Ähnlich wie bei anderen Aspekten virtueller Teams unterscheiden sich diese weniger grundsätzlich als eher graduell von traditionellen Teams. Daher ist ein gewisser Transfer von virtuellen Teams auch hinsichtlich Konfliktmanagementstrategien möglich. Der Erfolg virtueller Teams wird jedoch wesentlich durch die Einführung eines Konfliktmanagement-Systems gesteigert, das auf virtuelle Kooperationsformen abgestimmt ist und den damit verbundenen Herausforderungen gerecht wird"* (Konradt & Hertel, 2002, S.108).

Wie in traditionellen Teams kann auch in virtuellen Teams das Problem der Zurechenbarkeit von Individual-Leistungen zu Teammitgliedern auftreten. Schlimmstenfalls entziehen sich einzelne Mitglieder komplett der Gruppenarbeit und damit ihrer Verantwortung, zum Erfolg der Gruppe beizutragen. Die Folge davon wird als soziales Bummeln bezeichnet (vgl. Zimbardo & Gerrig, 2000, S.725; (Döring, 2003, S.498)). Innerhalb eines virtuellen Teams sollte Anonymität daher vermieden werden: *„Anonymität bei der computervermittelten Kommunikation sollte im Rahmen von CSCL unbedingt vermieden werden, um Trittbrettfahren und soziales Faulenzen zu vermindern. Auch wenn kurzfristig soziale Hemmungen überwunden werden, wenn sich Gruppenmitglieder anonym äußern können, trägt dies in einer Lerngruppe langfristig nicht dazu bei, dass sich eine vertrauensvolle Atmosphäre unter den Teilnehmenden entwickelt und so Ängste nachhaltig abgebaut werden können"* (Haake et al., 2004, S.47). Diese für Lerngruppen getroffene Feststellung lässt sich auch auf Arbeitsgruppen übertragen.

Trotzdem stellt die Beachtung der beiden letztgenannten Punkte keinen Garant für eine erfolgreiche Gruppenarbeit in virtuellen Teams dar, solange nicht die grundlegenden Gruppenprozesse (Kommunikation, Koordination und Kooperation; s. Kap.2.2) angemessene Unterstützung finden. So erläutert Döring (2003) die Problematik: *„Kollaboration im Netz, bei der die Beteiligten räumlich voneinander getrennt sind, sich möglicherweise nicht persönlich kennen und auch asynchron miteinander kommunizieren, stellt besonders hohe Anforderungen an die Koordination des Prozesses: Wissen und Fähigkeiten der Beteiligten müssen eruiert, Aufgaben verteilt und in ihrer zeitlichen Abwicklung aufeinander abgestimmt, Motivation und Commitment gefördert werden usw."* (Döring, 2003, S.284). Zur Unterstützung dieser Prozesse kommen spezielle (Software-)Systeme zum Einsatz, die in den nachfolgenden Kapiteln zusammen mit ihren zugehörigen Forschungsgebieten Erwähnung finden sollen.

3.2.3.3 Anforderungen an virtuelle Teams

Um den potentiellen Problemen, die virtuelle Arbeitsformen mit sich bringen, wirksam begegnen zu können, werden an virtuelle Teams und deren Mitglieder besondere Anforderungen gestellt. Da die Nutzung von Computern und des Internets unumgänglich sind, sollten die Mitglieder virtueller Teams nicht nur eine geringe

Angst vor dem Umgang mit Computern besitzen, sondern besser noch Freude am Arbeiten mit neuen Informations- und Kommunikationstechnologien besitzen.

Darüber hinaus sind Fähigkeiten wie die Bereitschaft zur Übernahme von Verantwortlichkeit, ein hoher Grad an Selbstorganisation, ausgeprägte kommunikative Eigenschaften auch bei der Nutzung elektronischer Medien sowie Teamorientierung und Konfliktfähigkeit trotz hoher Autonomie wünschenswerte Merkmale (vgl. Konradt & Hertel, 2002, S.26,S.53f).

Letztlich müssen die Mitglieder virtueller Teams auch bereit und in der Lage sein, den bereits genannten Organisationsaufwand mit zu tragen. Aufgrund der extrinsischen Motivation, die der Bildung virtueller Teams im Allgemeinen wie auch im speziellen Kontext der vorliegenden Arbeit zugrunde liegt, ist dieser Punkt jedoch weitgehend vernachlässigbar.

3.3 Zusammenfassung

Das Internet und seine Dienste bilden die unverzichtbare technische Grundlage des VitaminL-Systems als einer Plattform für die gemeinsame, zeitgleiche Java-Programmierung in virtuellen Teams. Die Kommunikation der Teilnehmer findet text-basiert in Form eines Chat statt, wobei zunächst eine strukturierte Dialogschnittstelle mit einer fest-definierten Menge vorgegebener Kommunikationselemente auf der Grundlage der *Collaborative Learning Skills* (CLS) realisiert werden wird (s. Kap.3.1.4). Mittels dieses Werkzeugs kann auf ein Parsing von natürlichsprachlichen Kommunikationsbeiträgen, das nicht im Fokus der vorliegenden Arbeit liegt, verzichtet werden.

Die Teilnehmer einer (Arbeits- oder Lern-)Gruppe, die unter ausschließlicher Verwendung von computer-vermittelter Kommunikation zusammenarbeiten, erfahren dabei eine Virtualisierung: Aus einer Gruppe wird eine virtuelle Gruppe beziehungsweise ein virtuelles Team (s. Kap.3.2). Die virtuelle Zusammenarbeit unterscheidet sich in vielen Punkten nicht wesentlich von traditioneller Teamarbeit, dennoch beinhaltet sie ein gewisses Problempotential (s. Kap.3.2.3.2), bietet aber auch Chancen (s. Kap.3.2.3.1), insbesondere durch Möglichkeiten, die aus der räumlichen und zeitlichen Entkopplung resultieren.

Kapitel 4

CSCW

4.1 Historie

Die computer-basierte Unterstützung von Arbeitsgruppen wird im interdisziplinären Forschungsgebiet CSCW (*Computer Supported Cooperative Work*) genauer untersucht: *„Das Forschungsgebiet Computer Supported Cooperative Work befasst sich mit der Rechnerunterstützung kooperativen Arbeitens. Ziel ist es, die Zusammenarbeit von Menschen durch den Einsatz von Informations- und Kommunikationstechnik zu verbessern, d.h. effizienter und flexibler, aber auch humaner und sozialer zu gestalten"* (Hasenkamp et al., 1994, S.15). Obwohl Engelbart (1962) mit seinem *Augment System* erste Ideen zur Unterstützung von Gruppenarbeit durch Computer veröffentlichte, die den Grundstock für nachfolgende Arbeiten in diesem Bereich bilden, dauerte es zwei Jahrzehnte bis zur endgültigen Etablierung des Begriffs CSCW und des zugehörigen eigenständigen Forschungsgebiets: 1984 wurde ein Workshop mit dem Titel *Computer Supported Cooperative Work (CSCW)* organisiert und 1986 wurde in Austin die erste CSCW-Konferenz abgehalten. Es folgten seitdem regelmäßig weitere CSCW-Konferenzen, zunächst in den Vereinigten Staaten, ab 1989 auch in Europa (vgl. Teufel et al., 1995, S.17f).

4.2 Forschungsdisziplinen

Obwohl CSCW sich längst als eigenständiges Forschungsgebiet etabliert hat, sind die Untersuchungen zur computergestützten Gruppenarbeit nicht auf eine einzige Disziplin beschränkt: *„CSCW-Forschung ist interdisziplinär. Auf CSCW-Konferenzen treffen Informatiker auf Sozialwissenschaftler wie Psychologen und Soziologen, auf Arbeitswissenschaftler und Designer, auf Ökonomen und Wirtschaftsinformatiker und viele andere Wissenschaftler"* (Schwabe et al., 2001, S.2). So kann CSCW je nach Blickwinkel als Teilgebiet der Informatik, der Psychologie, der Soziologie oder auch weiterer Organisations- und Sozialwissenschaften verstanden werden.

Die Sicht der Informatik entspricht häufig einer eher technisch orientierten Sicht: *„Im Vordergrund steht dabei zunächst die Diskussion, welche Methoden und Konzepte die Informatik anzubieten hat, die erfolgsversprechend im Zusammenhang mit der Entwicklung und dem Einsatz von CSCW-Werkzeugen genutzt werden könnten. Es wird aber auch angesprochen, welche spezifischen Aspekte, Eigenschaften und Schwächen noch abzudecken und zu lösen sind, bevor eine aus der Sicht der Informatik ausreichende Unterstützung bei der Entwicklung und dem Einsatz von CSCW-Werkzeugen gegeben ist"* (Schwabe et al., 2001, S.3).

Ergänzend dazu verfolgen andere Disziplinen eher am Menschen als an der Technik orientierte Ziele: *„Weitere beteiligte Disziplinen, die nicht zur Informatik zählen, stammen vorwiegend aus dem Gebiet der Soziologie und Psychologie. Wichtig ist die Sozialpsychologie, die sich als Teilgebiet der angewandten Psychologie mit Gruppenaspekten auseinandersetzt. So stammen viele empirische Arbeiten zur Untersuchung der Auswirkungen von CSCW-Applikationen von Sozialpsychologen"* (Teufel et al., 1995, S.20).

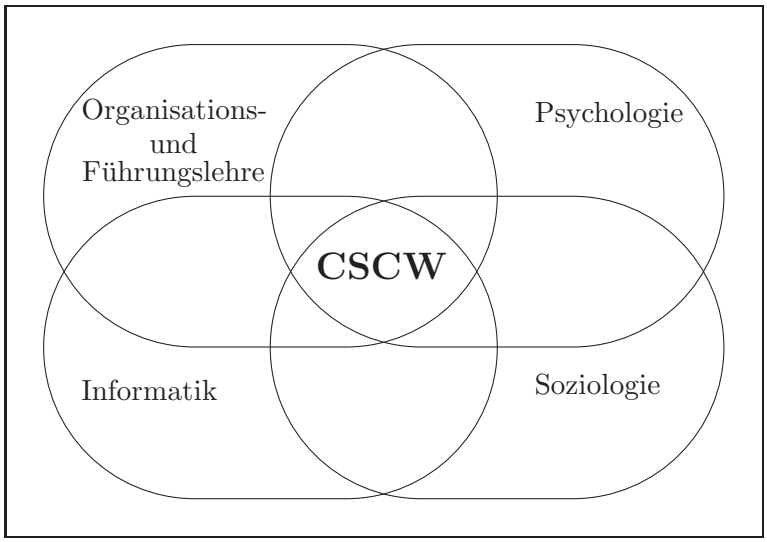

Abb. 4.1: Die Interdisziplinarität von CSCW
(vgl. Teufel et al., 1995, S.19)

Insgesamt lässt sich CSCW also im Schnittpunkt der genannten Disziplinen, namentlich der Informatik, der Soziologie, der Psychologie sowie den Organisations- und Führungslehren einordnen. Die Abbildung 4.1 illustriert diesen Sachverhalt.

4.3 Forschungsinhalte

Entsprechend der jeweils zugrundeliegenden Disziplin ergeben sich spezifische Sichtweisen auf CSCW, die wiederum mit eigenständigen Fragestellungen gekoppelt sind. So steuert die Sozialpsychologie im Rahmen der Kleingruppenforschung (s. Kap.2.1.2.2) Untersuchungen zum Einfluss der Gruppe auf den Einzelnen (und umgekehrt), zu Gruppengröße und Gruppenstrukturen – insbesondere Kommuni-

kationsstrukturen (s. Kap.2.3.3) und Rollenstrukturen (s. Kap.2.3.4) – sowie deren Auswirkungen auf die sinnvolle Gestaltung von CSCW-Systemen bei (vgl. TEUFEL ET AL., 1995, S.33; SCHWABE ET AL., 2001, S.33ff).

Aus der Organisationstheorie fliessen Beiträge in CSCW ein, die durch das grundlegende Organisationsproblem ausgelöst worden sind: *„Organisation begegnet uns auf Schritt und Tritt; denn Organisation ist immer dann notwendig, wenn Aufgaben zu bewältigen sind, die nicht von einer Person in einem Schritt erledigt werden können. Tatsächlich überschreiten die meisten Aufgaben, mit denen Menschen konfrontiert werden, die begrenzte Kapazität eines einzelnen Individuums. Ihre Bewältigung verlangt folglich nach Organisation. Das ist die Wurzel des Organisationsproblems"* (SCHWABE ET AL., 2001, S.76). Vereinfacht ausgedrückt beschäftigen sich die resultierenden Fragestellungen mit den Teilproblemen, eine Aufgabe in angemessene Teilaufgaben zu zerlegen und die Durchführung der identifizierten Teilaufgaben unter den Beteiligten zu koordinieren – und zwar unter Einsatz von Diensten computer-vermittelter Kommunikation (s. Kap.3.1): *„Die Analyse und Beschreibung von Gruppenaufgaben ist folglich eine wichtige Aufgabe der CSCW-Forschung"* (TEUFEL ET AL., 1995, S.33).

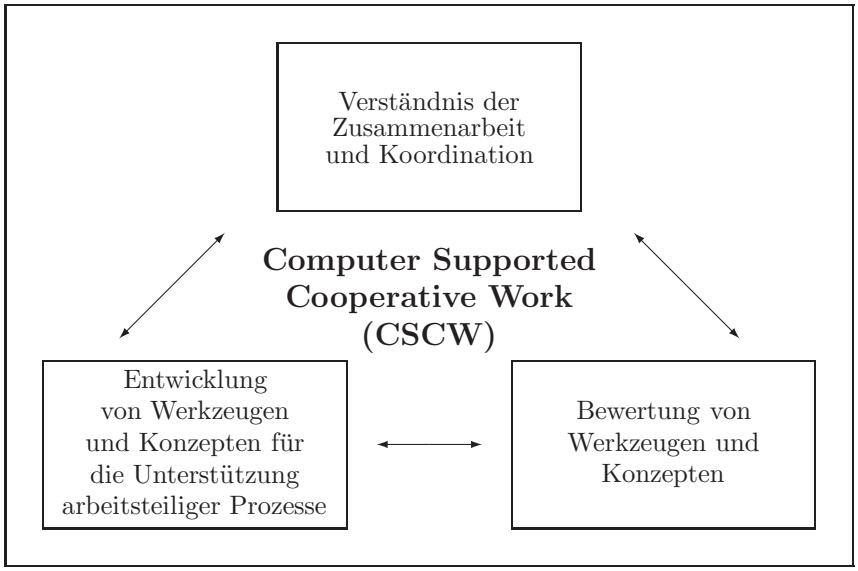

Abb. 4.2: Das CSCW-Forschungsgebiet
(vgl. HASENKAMP ET AL., 1994, S.16)

Neben weiteren Forschungsgebieten (wie bspw. Arbeitswissenschaft, Ethnographie oder auch Medientheorien) steuert letztlich auch die Informatik einen umfassenden Beitrag zum Thema CSCW bei. Abgesehen von wenigen spezialisierten Hardware-Komponenten handelt es sich dabei vornehmlich um Software, also CSCW-Applikationen (vgl. TEUFEL ET AL., 1995, S.35; SCHWABE ET AL., 2001, S.87ff) und damit verbunden Lösungsansätzen aus der Informatik und ihren Teildisziplinen, die ihren Beitrag zur Beantwortung anderweitig gestellter Fragen beitragen. Zu den betroffenen Teildisziplinen der Informatik gehören – neben den unverzichtbaren Grundlagen aus der Kommunikationstechnologie – beispielsweise Software-Ergonomie, Software-Entwicklung, Datensicherheit und Informationssys-

teme (vgl. SCHWABE ET AL., 2001, S.3).

HASENKAMP ET AL. (1994) entkoppeln diese unterschiedlichen Fragen- und Themenkomplexe von ihren Ausgangsdisziplinen und fassen sie in drei eng zusammenhängenden Themenbereichen zusammen, die *„die Entwicklung eines Verständnisses der Zusammenarbeit und Koordination, Konzepte und Werkzeuge zu deren Unterstützung sowie die Bewertung dieser Konzepte und Werkzeuge"* (HASENKAMP ET AL., 1994, S.16) als inhaltliche Schwerpunkte besitzen und in deren Mittelpunkt CSCW als eigenständiges Forschungsgebiet wiederzufinden ist (s. Abb.4.2).

4.4 Begriffsdiskussion

Entsprechend der vielen unterschiedlichen Blickwinkel und beteiligten Disziplinen existiert keine einheitliche Definition von CSCW, sondern eine Vielzahl zum Teil ergänzender, aber auch konkurrierender Begriffsbestimmungen. Mangels einer allgemein anerkannten Definition werden auch Diskussionen zu Teilbegriffen *Computer Support* (CS) und *Cooperative Work* (CW) geführt, was aber kaum zu einem besseren Verständnis führt (vgl. TEUFEL ET AL., 1995, S.14ff).

TEUFEL ET AL. (1995) schlagen daher folgende Definition von CSCW vor: *„Computer Supported Cooperative Work (CSCW) ist die Bezeichnung des Forschungsgebiets, welches auf interdisziplinärer Basis untersucht, wie Individuen in Arbeitsgruppen oder Teams zusammenarbeiten und wie sie dabei durch Informations- und Kommunikationstechnologie unterstützt werden können. Ziel aller Bemühungen im Gebiet CSCW ist es, unter Verwendung aller zur Verfügung stehenden Mittel der Informations- und Kommunikationstechnologie, Gruppenprozesse zu unterstützen und dabei die Effektivität und Effizienz der Gruppenarbeit zu erhöhen"* (TEUFEL ET AL., 1995, S.17).

Diese Definition findet auch in der vorliegenden Arbeit Anwendung, da sie gleichsam die Interdisziplinarität wie den Zugriff auf die Informatik und deren Werkzeuge in sich vereint.

4.5 Der Groupware-Begriff

Neben weiteren Begriffen, die im Zusammenhang mit CSCW oder gar synonym dazu verwendet werden (vgl. TEUFEL ET AL., 1995, S.16), wird immer wieder der Begriff *Groupware* genannt. Auch wenn CSCW und Groupware oftmals in der Literatur gleich gesetzt werden, so haben diese doch unterschiedliche Bedeutungen: CSCW umfasst das Forschungsgebiet, das sich umfassend mit den Aspekten computer-unterstützter Gruppenarbeit befasst, Groupware bezeichnet daraus resultierende Produkte (CSCW-Systeme), wobei sich eine eher technologisch orientierte Sichtweise durchgesetzt hat. Genaugenommen handelt es sich bei Groupware um ein Kunstwort, zusammengesetzt aus *Group* und *Software*, also Software

für Gruppen. Dies entspricht auch im Kern der Groupware-Definition von Teufel et al. (1995): *„Groupware bzw. CSCW-Applikationen sind aus Software und eventuell spezifischer Hardware bestehende Systeme, durch die Gruppenarbeit unterstützt oder ermöglicht wird"* (Teufel et al., 1995, S.22).

Groupware entsteht aus der sinnvollen Kombination der in Kapitel 3.1 vorgestellten Internet-Dienste: *„Wenn in Arbeitsgruppen computervermittelt kommuniziert und kollaboriert wird, dann greift man in Unternehmen meist auf* Groupware *bzw. CSCW-Systeme (*Computer Supported Cooperative Work*) zurück. Diese erlauben eine stärkere Strukturierung und Kontrolle der Gruppenprozesse als einfache Internet-Dienste"* (Döring, 2003, S.512). Zunächst sollen jedoch zwei grundlegende Einsatzkonzepte für CSCW betrachtet werden, bevor die Grundbausteine von Groupware näher erläutert werden.

4.5.1 Workflow Management

Ein *Workflow* (dt.: Arbeitsablauf) stellt (stark vereinfacht) eine Menge von Aktivitäten in Beziehung zueinander mit dem Zwecke, betriebliche, technische oder verwaltungstechnische Abläufe analysieren, dokumentieren und modellieren zu können. Oder wie es Teufel et al. (1995) definieren: *„Ein Workflow ist eine endliche Folge von Aktivitäten, wobei die Folge durch Ereignisse ausgelöst und beendet wird (Begriffe wie Business Process, Vorgangskette, Prozesskette, Geschäftsprozess oder Geschäftsvorgang werden hier als Synonyme des Begriffs* Workflow *verstanden)"* (Teufel et al., 1995, S.182).

Workflow Management beinhaltet alle Tätigkeiten, die für einen funktionierenden Workflow notwendig sind: *„Workflow Management umfasst alle Aufgaben, die bei der Modellierung, der Simulation sowie bei der Ausführung und Steuerung von Workflows erfüllt werden müssen"* (Teufel et al., 1995, S.182). Dies beinhaltet schwerpunktmäßig unternehmensweite Abläufe mit einer Vielzahl von Teilnehmern. Die unterstützten betrieblichen Abläufe weisen in der Regel einen hohen Strukturierungsgrad, eine geringe Komplexität und eine hohe Wiederholungsfrequenz auf (vgl. Hasenkamp et al., 1994, S.26; Teufel et al., 1995, S.183).

4.5.2 Workgroup Computing

Beim *Workgroup Computing* werden – im Gegensatz zum *Workflow Management* und dessen Schwerpunkt auf unternehmensweite Abläufe – zumeist kleine, überschaubare Gruppen betrachtet, deren zu bearbeitende Aufgaben oftmals nur gering strukturiert sind (vgl. Teufel et al., 1995, S.209). Da die Aufgaben und Abläufe im Bereich des Workgroup Computing ferner nur einen sehr geringen Wiederholungsgrad aufweisen, werden an die entsprechenden Werkzeuge zur Unterstützung von Workgroup Computing besonders hohe Anforderungen hinsichtlich Flexibilität gestellt. Im Mittelpunkt der Betrachtungen beim Workgroup Computing stehen Arbeitsgruppen und die Zusammenarbeit innerhalb diesen Gruppen, wohingegen

das Workflow Management auf der Ebene der Gesamtorganisation anzusiedeln ist (vgl. HASENKAMP ET AL., 1994, S.26). Die nachfolgende Tabelle 4.1 stellt die beiden grundlegenden Einsatzkonzepte von Groupware anhand charakteristischer Kriterien gegenüber.

Tab. 4.1: Workflow Management und Workgroup Computing

	Workflow Management	**Workgroup Computing**
Koordinationsmodell	≫Aufteilung und Lösung von Teilproblemen≪	≫Lösung *eines* gemeinsamen Problems≪
Anzahl der Beteiligten	hoch	niedrig
Räumliche Verteilung der Beteiligten	an einem Ort / an verschiedenen Orten	an einem Ort / an verschiedenen Orten
Zeitliche Verteilung	*bisher:* zu unterschiedlichen Zeiten	zur gleichen Zeit / zu unterschiedlichen Zeiten
Strukturierungsgrad der Aufgabe(n)	*bisher:* hoch	mittel / gering
Wiederholungsfrequenz	*bisher:* hoch	mittel / gering
Bedeutung organisatorischer Regeln	hoch	niedrig
≫Organisatorischer Bezug≪	Organisationsweite Prozesse	Gruppe
Einbindung in Gesamtorganisation	ja	*bisher:* gering
Anbindung an betriebliche Informationsverarbeitung	zum Teil	*bisher:* nein
Primäres Ziel	*bisher:* Effizienz	*bisher:* Flexibilität
Aktive Steuerung und Verfolgung des Arbeitsschrittes	ja	*bisher:* nein

(HASENKAMP ET AL., 1994, S.27)

4.6 Elemente von Groupware

Groupware zeichnet sich nicht durch einen fest definierten Funktionsumfang aus, sondern stellt eher eine Klasse bestimmter Applikationen dar, die sich aus einer Menge möglicher Dienste und Elemente zur Unterstützung von Gruppenarbeit bedienen. Es folgt eine kurze Betrachtung der wichtigsten Groupware-Elemente, klassifiziert nach den jeweils von ihnen unterstützten Gruppenprozessen Kommunikation, Koordination und Kooperation (s. Kap.2.2.1, Kap.2.2.2 und Kap.2.2.3),

wobei – wie schon bei den in Kapitel 3.1 genannten Internet-Diensten – anzumerken ist, dass eine trennscharfe Kategorisierung nicht immer möglich ist.

4.6.1 Wahrnehmung von Gruppenmitgliedern

Der gegenseitigen Wahrnehmung der Mitglieder eines virtuellen Teams kommt eine besondere Bedeutung zu, weil ohne diese Wahrnehmung eine wichtige Grundlage zur Kommunikation und darauf aufbauend zu Koordination und Kooperation fehlt: *„Bei der gemeinsamen Arbeit ist das »Gewahrsein« (*awareness*) der Anwesenheit und des Verhaltens anderer Gruppenmitglieder eine Grundvoraussetzung für gemeinsames und koordiniertes Handeln"* (Holmer et al., 2001, S.184).

So ist es für den Einzelnen zunächst von Bedeutung, überhaupt über die Anwesenheit weiterer Gruppenmitglieder informiert zu sein. Darüber hinaus hat es sich – wie in traditionellen Kooperationsformen, bei denen die Teilnehmer sich am selben Ort befinden – als hilfreich erwiesen, weitere Information über den Zustand der Gruppenmitglieder zu erhalten. In der Virtualität beinhaltet dies Information über aktuelle Tätigkeiten (bspw. Navigation innerhalb der Kooperationsumgebung, Bearbeitung eines oder mehrerer Dokumente), kann darüber hinaus aber auch Gemütszustände (wie gelangweilt, gereizt, erheitert etc.) beinhalten und ferner Absichten oder Ziele der Kooperationspartner (vgl. Holmer et al., 2001, S.184; Pfister & Wessner, 2001b, S.258; Holmer & Jödick, 2004, S.88).

Im Falle asynchroner Kooperationen sollten den Mitglieder im Rahmen von Awareness-Funktionalitäten auch Informationen über vergangene, für die Gruppenarbeit relevante Ereignisse wie bearbeitete Dokumente und erledigte Teilaufgaben bereitgestellt werden. Diese Informationen können darüber hinaus auch von Tutoren verwendet werden, um den Gruppenfortschritt zu überblicken und gegebenenfalls korrigierend eingreifen zu können (Holmer & Jödick, 2004, S.88).

Die Umsetzung von Awareness-Funktionen kann somit durch verschiedene Elemente wie beispielsweise *„durch die Anzeige von Historien oder die Auflistung aktueller Aktivitäten"* (Koch, 2001, S.290) erfolgen. Im einfachsten Falle wird in einem CSCW-System eine Liste der derzeit angemeldeten Benutzer angezeigt. Somit sind zumindest alle potentiellen Kommunikationspartner bekannt.

4.6.2 Kommunikationsorientierte Elemente

Werkzeuge, die zur Gruppenkommunikation verwendet werden können, werden üblicherweise weiter unterteilt anhand des zeitlichen Versatzes zwischen einzelnen Botschaften in asynchrone und synchrone Kommunikationswerkzeuge (vgl. Döring, 2003, Kap.2.2f). Dabei können verschiedene Konzepte zur Anwendung kommen: *„Die Kommunikation zwischen Akteuren kann auf zwei unterschiedlichen Verfahren beruhen ...: dem expliziten Nachrichtenaustausch (message passing) oder der impliziten Kommunikation über gemeinsame Arbeitsbereiche beziehungsweise Datenstrukturen (z.B. ein Datenbank- oder Blackboard-System)"* (Ha-

senkamp et al., 1994, S.23). Diese grundlegenden Kommunikationsmodelle sind in der nachfolgenden Abbildung dargestellt.

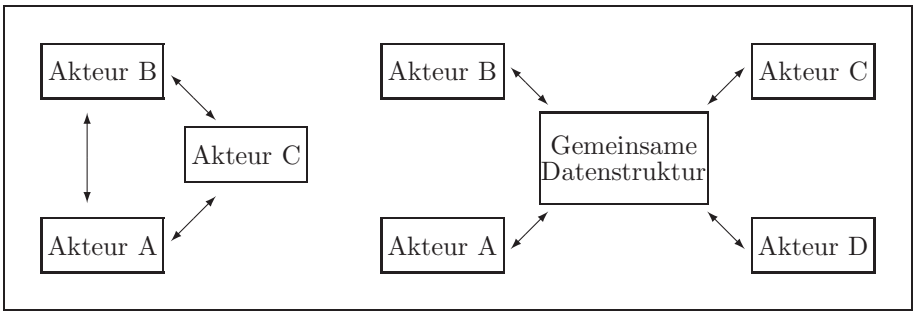

Abb. 4.3: Grundlegende Kommunikationsmodelle
(vgl. Hasenkamp et al., 1994, S.23)

Zu denjenigen Werkzeugen, die zur asynchronen Kommunikation verwendet werden, zählen typischerweise E-Mail-Systeme, Bulletin Board-Systeme und Newsgroups, wohingegen für synchrone Kommunikation üblicherweise Konferenzsysteme (vom Chat bis zur Videokonferenz) zum Einsatz kommen.

4.6.2.1 E-Mail

Dieser Dienst regelt den Austausch von elektronischer Post. Obwohl er ursprünglich rein textbasiert war, können inzwischen auch multimedial gestaltete Nachrichten zeitversetzt zwischen Sender und Empfänger ausgetauscht werden. Ferner ist das Beifügen von Dokumenten jeglicher Art als Anhang möglich. Soll E-Mail im Rahmen von Gruppenarbeit sinnvolle Verwendung finden, so ist der dazu notwendige Aufwand zu berücksichtigen, da beispielsweise jeder Teilnehmer seine eigene Adressatenliste lokal pflegen muss. Insofern kann E-Mail in diesem Kontext lediglich als Basisdienst zur asynchronen Kommunikation zwischen den Mitgliedern einer Arbeitsgruppe betrachtet werden (vgl. Hasenkamp et al., 1994, S.28f; Teufel et al., 1995, S.130ff; Schwabe et al., 2001, S.168ff; Döring, 2003, S.50ff; Haake et al., 2004, S.67f).

4.6.2.2 Bulletin Board-Systeme

Wenn man E-Mails als Produkt der Virtualisierung von Briefpost interpretiert, dann sind Bulletin Board-Systeme das digitale Gegenstück zu Pinwänden oder schwarzen Brettern, auf die nur ein ausgewählter Personenkreis Zugriff hat. Sie *„bieten die Möglichkeit, Gruppen einzurichten. Gruppenmitglieder können Beiträge abgeben und die Beiträge aller anderen Gruppenmitglieder lesen. Bilaterale Kommunikation ist ebenfalls möglich"* (Hasenkamp et al., 1994, S.170). Kernstück ist in der Regel eine Datenbank oder eine sonstige zentrale Verwaltungseinheit, das heißt diese Systeme werden in der Regel mit einem Server realisiert. Der Server nimmt dabei die Rolle eines Gruppengedächtnisses ein, da er alle Beiträge beliebig lange aufbewahren kann und auch eine spätere Nutzung von Beiträgen

ermöglicht: *„Das Konferenzsystem speichert die Nachrichten und garantiert, dass nur Gruppenmitglieder Nachrichten der Gruppe lesen und an sie schreiben können. Die Speicherung der Nachrichten im Konferenzsystem erlaubt die Reorganisation der Nachrichten und das Nachvollziehen der bisherigen Kommunikation"* (Haake et al., 2004, S.69). Für die Gruppenarbeit sind diese Systeme insofern von Nutzen, als dass eine zentrale Speicherung (und Sicherung) der gesamten Gruppenkommunikation ermöglicht wird. Vor allem kann bei konsequenter Nutzung eines solchen schwarzen Bretts gewährleistet werden, dass alle Mitglieder einer Gruppe – und nur die Mitglieder – stets vollen Zugriff auf die gesamte Gruppenkommunikation (auch im Rückblick) besitzen (vgl. Schwabe et al., 2001, S.170f; Haake et al., 2004, S.68f).

Newsgroups funktionieren nach einem ähnlichen Prinzip wie die soeben genannten Bulletin Board-Systeme, sind aber traditionell eher öffentlich zugänglich (vgl. Schwabe et al., 2001, S.171f; Haake et al., 2004, S.69f). Einer Gleichsetzung beider Systemarten wie beispielsweise in (Teufel et al., 1995, S.154f) und (Döring, 2003, S.62ff) kann sich der Autor der vorliegenden Arbeit aus eben diesem Grund nicht anschließen. Aufgrund der öffentlichen, teilweise sogar anonymen Nutzung sind Newsgroups für die vorliegende Arbeit weniger relevant und werden daher nicht weiter ausgeführt.

4.6.2.3 Konferenzsysteme

Die Realisierung von synchroner Kommunikation steht als Anwendungsszenario bei den Konferenzsystemen im Vordergrund: *„Sie unterstützen eine genau adressierbare Menge von Kommunikationspartnern, die zur gleichen Zeit miteinander kommunizieren wollen. Die Unterstützung besteht vor allem in der Raumüberbrückung. Konferenzsysteme können mit verschiedenen Medien realisiert werden. So gibt es rein textuelle, audiobasierte oder videobasierte Konferenzsysteme sowie Mischformen"* (Teufel et al., 1995, S.142f).

Die textbasierten Konferenzsysteme zählen zu den einfachsten und auch frühesten Systemen dieser Kategorie. Typische Vertreter sind die verschiedenen Chat-Varianten, bei denen zwei oder mehr Teilnehmer zeitgleich textbasierte Nachrichten von wenigen Worten beziehungsweise wenigen Textzeilen austauschen (vgl. Teufel et al., 1995, S.143; Schwabe et al., 2001, S.160; Döring, 2003, S.83ff; Haake et al., 2004, S.73ff).

Aufgrund technischer Fortschritte bei Bandbreiten, Übertragungs- und Verarbeitungsgeschwindigkeiten können videobasierte Echtzeitübertragungen inzwischen auch von Standard-Desktop-PCs durchgeführt werden. Vorteile gegenüber der rein textbasierten Synchronkommunikation ergeben sich, da die übertragenen Informationen wesentlich detailierter und situationsbezogener sind, da Sendern wie Empfängern mehr Kanäle für die Übertragung von Informationen zur Verfügung stehen: *„Videokonferenzsysteme unterstützen eine nonverbale Kommunikation mit Hilfe von Gestik und Mimik. Sie liefern auch viele Kontextinformationen, da alle*

Objekte, Personen und Ereignisse im Aufnahmebereich der Kamera sichtbar sind" (Schwabe et al., 2001, S.161).

Ergänzend sind auch audiobasierte Konferenzen mittels Telefon oder Internet-Telefonie sowie Mischformen zwischen den drei genannten Grundformen Text, Audio und Video möglich, auf eine vertiefende Betrachtung soll an dieser Stelle jedoch mangels Relevanz für die vorliegende Arbeit verzichtet werden, stattdessen wird der interessierte Leser auf die entsprechende Literatur verwiesen (vgl. Teufel et al., 1995, S.142ff; Schwabe et al., 2001, S.160ff; Döring, 2003, S.82,S.97f; Haake et al., 2004, S.77f).

Synchrone Kommunikation (in Form von Konferenzen) ist im Rahmen von Gruppenarbeit beispielsweise zum Treffen von Entscheidungen sinnvoll, insbesondere wenn sie textbasiert stattfindet: *„In weltweit verteilten Teams wird sie für Abstimmungen genutzt. Deren Verlauf und Ergebnis bleiben für Kollegen nachvollziehbar, die wegen der Zeitverschiebung nicht teilnehmen konnten"* (Schwabe et al., 2001, S.163). Bei komplexen Aufgaben hat sich zwar die videovermittelte Kommunikation als vorteilhaft gegenüber den anderen Formen gezeigt (vgl. Schwabe et al., 2001, S.166), jedoch ist der Aufwand (technisch wie organisatorisch) bei der Realisierung und Durchführung ungleich höher als beispielsweise Chat-Kommunikation. Die Verwendung textbasierter Kommunikation hat auch Nachteile: Durch Verengung der Kommunikation auf einen einzigen Kanal werden die Ausdrucksmöglichkeiten erheblich eingeschränkt, was – gerade in größeren Gruppen – zu Koordinationsproblemen führen kann (vgl. Haake et al., 2004, S.75).

4.6.3 Koordinationsorientierte Elemente

Wie bereits in Kapitel 2.2.2 ausgeführt, ist eine aufeinander abgestimmte Ausführung aller interdependenten Gruppenaktivitäten für eine erfolgreiche Erreichung des Gruppenziels unerlässlich. Dazu werden im Rahmen von Groupware und CSCW spezielle Werkzeuge angeboten: *„Koordinationswerkzeuge erleichtern oder ermöglichen die Handhabung von Abhängigkeit. Durch die räumliche und zeitliche Verteilung der beteiligten Personen steigen die Koordinationsprobleme und Koordinationswerkzeuge gewinnen an Bedeutung"* (Schwabe et al., 2001, S.174).

Koordinationsbedarf bei der computer-basierten Gruppenarbeit entsteht einerseits aufgrund der räumlichen Verteilung der Beteiligten sowie aufgrund der Notwendigkeit, zu vereinbarten Zeitpunkten synchron zu kommunizieren. Ferner resultiert aus der Gruppenaufgabe, den aus ihr abgeleiteten Aktivitäten und zwischen diesen Aktivitäten existierenden Abhängigkeiten weiterer Koordinationsbedarf.

4.6.3.1 Gruppenterminkalender

Die zeitliche Koordination von Gruppenmitgliedern kann mittels elektronischer Gruppenterminkalender vereinfacht werden. Prinzipiell kann die Terminverwaltung

aller Gruppenmitglieder in einem zentralen Kalendersystem erfolgen: *„Diese Kalender stellen einer Gruppe eine gemeinsame Datenbank zur Verfügung. In dieser Datenbank kann jeder Berechtigte alle Termine seiner Kollegen anschauen. Es ist auch möglich, den Gruppenterminkalender einen freien Termin für eine Gruppe automatisch suchen zu lassen. Ein freier Termin kann dann auch automatisch in die Kalender der Betroffenen eingetragen werden"* (Schwabe et al., 2001, S.175).

Diese Offenheit ist aber nicht immer von allen Teilnehmern erwünscht. Auch rechtliche Probleme können sich (bspw. aufgrund fehlenden Datenschutzes) aus dieser Vorgehensweise ergeben, da sämtliche Termine für alle Mitglieder der Arbeitsgruppe einsehbar sind (vgl. Teufel et al., 1995, S.211; Schwabe et al., 2001, S.175). Mittels entsprechender Berechtigungskonzepte kann Abhilfe geschaffen werden.

Als weiterer Ansatz ist ein dezentrales Kalendersystem denkbar, in welchem an andere nur noch Terminvorschläge verschickt werden und deren Rückmeldungen (Akzeptanz oder Ablehnung eines Vorschlags) weiterverarbeitet werden. Hier ist jedoch ein möglicher höherer Aufwand zu berücksichtigen. Insgesamt aber muss festgestellt werden, dass bei der räumlich verteilten Zusammenarbeit in virtuellen Teams die Festlegung von Phasen synchroner Kommunikation mittels elektronischer Terminkalender vereinfacht werden kann, sobald die Gruppengröße eine bestimmte *kritische Masse* erreicht oder überschreitet.

4.6.3.2 Workflow-Management-Werkzeuge

Die Unterstützung von Abläufen auf Unternehmensebene (s. Kap.4.5.1) ist Ziel und Aufgabe von Workflow-Management-Systemen: *„Vorgänge werden nach vorher festgelegten Regeln auf einem festgelegten Pfad (gemäß dem Dienstweg) automatisch durch eine Organisation oder Behörde geleitet und den zuständigen Bearbeitern automatisch zugestellt"* (Schwabe et al., 2001, S.176). Ähnlich formulieren es auch Teufel et al. (1995): *„Workflow Management-Systeme dienen primär der Koordination organisationsweiter, arbeitsteiliger Prozesse. Sie ermöglichen die Zusammenarbeit nach vordefinierten Regeln und Methoden, wobei meist die gemeinsame Nutzung von Dokumenten im Vordergrund steht"* (Teufel et al., 1995, S.84).

Ein typisches Einsatzgebiet für Workflow Management ist die Büroautomation (vgl. Hasenkamp et al., 1994, S.26; Schwabe et al., 2001, S.176); typische Workflow-Management-Systeme zur Unterstützung dieses Aufgabengebiets besitzen also in der Regel ein Dokumentenmanagementsystem. In neueren Generationen von Workflow-Management-Systemen werden zunehmend weitere Funktionalitäten wie *Enterprise-Content-Management* (ECM; = Technologien zur Erfassung, Verwaltung, Speicherung, Bewahrung und Bereitstellung von Content und Dokumenten zur Unterstützung von organisatorischen Prozessen) oder auch *Enterprise Resource Planning* (ERP; = möglichst effiziente Einplanung aller in einem Unternehmen vorhandenen Ressourcen für den betrieblichen Ablauf) integriert (vgl. Teufel et al., 1995, S.182ff; Schwabe et al., 2001, S.205ff).

4.6.3.3 Projektmanagement-Werkzeuge

Die zentrale Koordination innerhalb kleiner Gruppen bei mittel bis gering strukturierten Aufgaben ist das typische Einsatzgeiet von Werkzeugen zur Projektplanung, -durchführung und -kontrolle. Die diesem Anwendungsfeld zuzuordnenden Systeme besitzen in der Regel Funktionalitäten, die

- eine Planung aller dem Projekt zugeordneten Ressourcen und Aktivitäten ermöglichen,
- die Verfolgung des Projektstatus unterstützen und
- Kommunikation zwischen allen Projektbeteiligten fördern.

Auch eine Anbindung an unternehmensweit verfügbare Instrumente (Berichtswesen, Budgetplanung etc.) ist sinnvoll und wünschenswert (vgl. Schwabe et al., 2001, S.176f).

Die Vielzahl existierender Produkte in diesem Bereich ist nahezu unübersichtlich, daher seien als typische Vertreter *MS Project*, *MindPlan Project*, *Openworkbench*,*PHProjekt* und *Project/Open* genannt. Für weitere Produkte und Informationen sei an dieser Stelle auf die einschlägige Literatur verwiesen.

4.6.3.4 Spezielle Datenbanken

Die Anforderungen, die CSCW-Systeme an zu verwendende Datenbanken zur Schaffung gemeinsamer Informationsräume stellen, gehen üblicherweise über den Funktionsumfang von Standard-Datenbanken hinaus, denn *„in vielen Fällen ist es notwendig, dass das Datenhaltungssystem spezifische Zugriffsrechte unterstützt, mit denen die verschiedenen Benutzerrollen ausgestattet werden können. Weitere Anforderungen betreffen die Unterstützung multimedialer bzw. komplexer Objekte (Objekte von Objekten), langer Transaktionen, flexibler Konsistenzprüfungen, von Vererbungshierarchien, Triggermechanismen und Datenverteilung etc."* (Teufel et al., 1995, S.174).

Mittels verschiedener Ansätze wird versucht, diesen Anforderungen gerecht zu werden und die damit verbundenen Probleme zu lösen. Ein vielversprechender Ansatz kann in der sog. *Middleware* gefunden werden: Middleware kann als Vermittler zwischen Anwendungen verstanden werden und beispielsweise als Transaktionsschicht für ansonsten voneinander unabhängige Teilkomponenten fungieren (vgl. Badach et al., 2003, S.322). Der Einsatz von Middleware-Konzepten und -Techniken macht somit aus mehreren eher einfachen Komponenten (Servern, Datenbanken) aus Sicht des Benutzers eine einzige, leistungsstarke Komponente, die auch den eingangs genannten Anforderungen von Groupware gerecht werden kann. Technisch handelt es sich dabei üblicherweise um 3-Schichten-Modelle anstelle der klassischen 2-Schichten-Modelle (Client-Server-Architekturen): Zwischen

die Client-Schicht und die Server-Schicht wird als vermittelnde Schicht die Applikationsschicht eingeführt, die Befehle von einem Client entgegennimmt, an Komponenten der Server-Schicht weiterleitete, deren Ergebnisse zusammenführt und das Gesamtergebnis an den anfragenden Client zurückliefert.

4.6.3.5 Verteilte Hypertext-Systeme

Hypertext-Systeme realisieren nicht-lineare Darstellungen von Informationen in Form von Texten und Graphiken. Im Gegensatz zu der in Büchern linearen Anordnung von Informationen ermöglichen Hypertexte die direkte Navigation zwischen verschiedenen Themen. Den *de facto* Standard bildet das Internet, genau genommen der als *World Wide Web* bekannte Teil, der auf dem `http`-Protokoll basiert und `HTML` als Beschreibungssprache verwendet (s. Kap.3.1.1; Teufel et al., 1995, S.164ff Badach et al., 2003, S.6f,S.65ff).

Der Grundgedanke beim Einsatz verteilter Hypertext-Systeme zur Unterstützung der Koordination liegt in der dezentralen Steuerung der Aktivitäten aller Beteiligten über ein gemeinsames Material (vgl. Schwabe et al., 2001, S.177f). Aufgrund seines Ursprungs, einen weltweit jederzeit zugänglichen Wissenspool zu schaffen (vgl. Badach et al., 2003, S.6f), stellt die Suche nach Informationen immer noch eines der Hauptanwendungsgebiete des WWW dar. Auch ist die in Teufel et al. (1995) noch fehlende Sicherheit (vgl. Teufel et al., 1995, S.173) durch zusätzliche Protokolle (wie bspw. `https`) und kryptologische Methoden (Verschlüsselungsverfahren, digitale Signaturen, Wasserzeichen etc.) inzwischen gewährleistet, so dass den Mitgliedern einer Arbeitsgruppe über das Internet beziehungsweise das WWW inzwischen auch sensible Daten bereit gestellt werden können. Dies können Nachschlagewerke wie Lexika oder Sammlungen von Produktbeziehungsweise Bauteilspezifikationen sein, aber auch interaktive Anwendungen, die auf spezielle Datenbank-Systeme zurückgreifen (s. Kap.4.6.3.4).

4.6.4 Kooperationsorientierte Werkzeuge

Die Unterstützung von Abläufen und Aktivitäten auf der Ebene von Arbeitsgruppen erfolgt im Rahmen von CSCW-Systemen meist auf der Basis gemeinsamer Materialien und deren Bearbeitung durch die Gruppe (vgl. Schwabe et al., 2001, S.181). Entsprechend einer Unterscheidung in synchrone und asynchrone Kooperationen und Kooperationswerkzeuge zählen dazu gemeinsame Objekte (vgl. Schwabe et al., 2001, S.181f) und gemeinsame Arbeitsbereiche (vgl. Schwabe et al., 2001, S.194f). Auch Systeme zur Unterstützung von Gruppensitzungen und -entscheidungen fördern die Zusammenarbeit auf Ebene der Arbeitsgruppe, obwohl einige der in diesem Kapitel genannten Werkzeuge auch auf Unternehmensebene und zu Koordinationszwecken anwendbar sind (s. Kap.4.6.3.2) und umgekehrt.

4.6.4.1 Sitzungsunterstützungssysteme

Bei der Durchführung computer-unterstützter Sitzungen sind – wie auch in traditionellen Sitzungen – die einzelnen Aktivitäten einer Sitzung (Tagesordnungspunkte, Redezeiten der Teilnehmer und die ihnen zugeordneten Werkzeuge etc.) vorab festzulegen. Im Verlauf einer Sitzung wird der zuvor erarbeitete Ablauf gesteuert (und im Bedarfsfall flexibel angepasst), so dass die Kernthemen der Sitzung stets im Mittelpunkt der Betrachtungen bleiben (vgl. Schwabe et al., 2001, S.179). Die Integration weiterer Groupware-Werkzeuge zur Erweiterung des Funktionsumfangs in Sitzungsunterstützungssystemen ist sicherlich wünschenswert (vgl. Teufel et al., 1995, S.230), jedoch ist der zu realisierende Funktionsumfang mit Bedacht an die tatsächlichen Bedürfnisse des jeweiligen Anwendungskontext anzupassen, da ansonsten eine Überforderung der Benutzer zu befürchten ist[1].

4.6.4.2 Entscheidungsunterstützungssysteme

Die Hauptaufgabe von Entscheidungsunterstützungssystemen liegt in der Unterstützung einer Entscheidungsfindung innerhalb einer Arbeitsgruppe: *„Entscheidungsunterstützungssysteme sind Systeme, welche die Effektivität von Entscheidungsprozessen von Gruppen im Rahmen teilweise strukturierbarer Aufgaben unterstützen. Ein Entscheidungsunterstützungssystem für Gruppen muss zwingend von mehreren Personen benutzbar sein bzw. benutzt werden"* (Teufel et al., 1995, S.227).

Oftmals sind diese Entscheidungsunterstützungssysteme in Sitzungsunterstützungssysteme eingebettet oder werden von diesen benutzt, um im Rahmen von Gruppensitzungen auch Entscheidungsfindungen durchführen zu können (vgl. Teufel et al., 1995, S.228).

4.6.4.3 Gruppeneditoren

Werkzeuge dieser Kategorie ermöglichen den Gruppenmitgliedern, gemeinsam Dokumente zu erstellen und zu bearbeiten. Je nach Werkzeug können in einem Dokument mehrere Benutzer gleichzeitig an unterschiedlichen Stellen arbeiten und Änderungen vornehmen beziehungsweise es besitzt nur ein Autor Schreibzugriff auf ein Dokument (und damit verbunden das Recht, Änderungen an dem Dokument durchzuführen), während die übrigen Teilnehmer zwar unmittelbar über diese Änderungen informiert werden, indem sie bei den Teilnehmern angezeigt werden, aber selbst nur Leserechte besitzen (vgl. Teufel et al., 1995, S.216f).

Die auf die Gruppenmitglieder verteilten Dokumente können dabei als gemeinsame Objekte (vgl. Schwabe et al., 2001, S.181f) verstanden werden, über deren Inhalte (in der Regel in Form von Text und Graphiken) Kooperationen innerhalb

[1] Diese Gefahr besteht im Prinzip bei fast jeder Art von Computer-Anwendung, sei es nun eine Textverarbeitung wie *MS Word*, eine Bildverarbeitung wie *Photoshop* oder eine Entwicklungsumgebung wie *eclipse*.

einer Gruppe ermöglicht werden. In der Informatik hat sich der Begriff der *Shared Documents* gebildet, der Mechanismus zum verteilten Bearbeiten wird *Document Sharing* genannt. Das Umfeld, in dem Kooperationen mittels gemeinsamer Objekte stattfinden kann, sind nach Schwabe et al. (2001) die gemeinsamen Arbeitsbereiche (vgl. Schwabe et al., 2001, S.194ff).

Diese Gruppeneditoren haben für die vorliegende Arbeit eine besondere Bedeutung, da sie als eines der Basiskonzepte zur Unterstützung der gemeinsamen Java-Programmierung in die Entwicklung der VitaminL-IDE eingeflossen sind (s. Kap.8.3).

4.7 Klassifikationen

Die Klassifikation von CSCW-Systemen beziehungsweise deren Elementen kann anhand unterschiedlicher Kriterien erfolgen. Zu den wichtigsten Kriterien gehören die zeitliche und die räumliche Verteilung der Teilnehmer, aus denen die in Abbildung 4.4 dargestellte Raum-Zeit-Matrix resultiert.

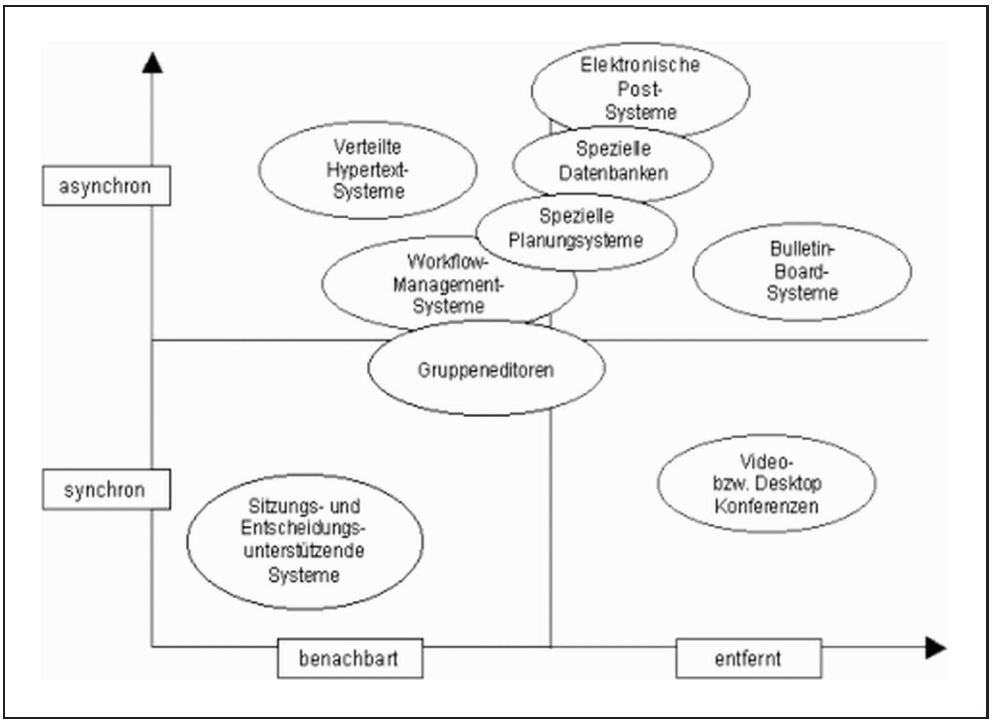

Abb. 4.4: Raum-Zeit-Matrix
(vgl. Teufel et al., 1995, S.25)

Die Klassifikation anhand nur zweier Dimensionen (Raum und Zeit) wird aufgrund ihrer nur groben Rasterung durchaus kritisiert (vgl. Teufel et al., 1995, S.25ff). Daher finden sich weitere Klassifikationen, etwa anhand der zugehörigen Unterstützungsfunktion hinsichtlich Kommunikation, Koordination und Kooperation (s. Abb.4.5).

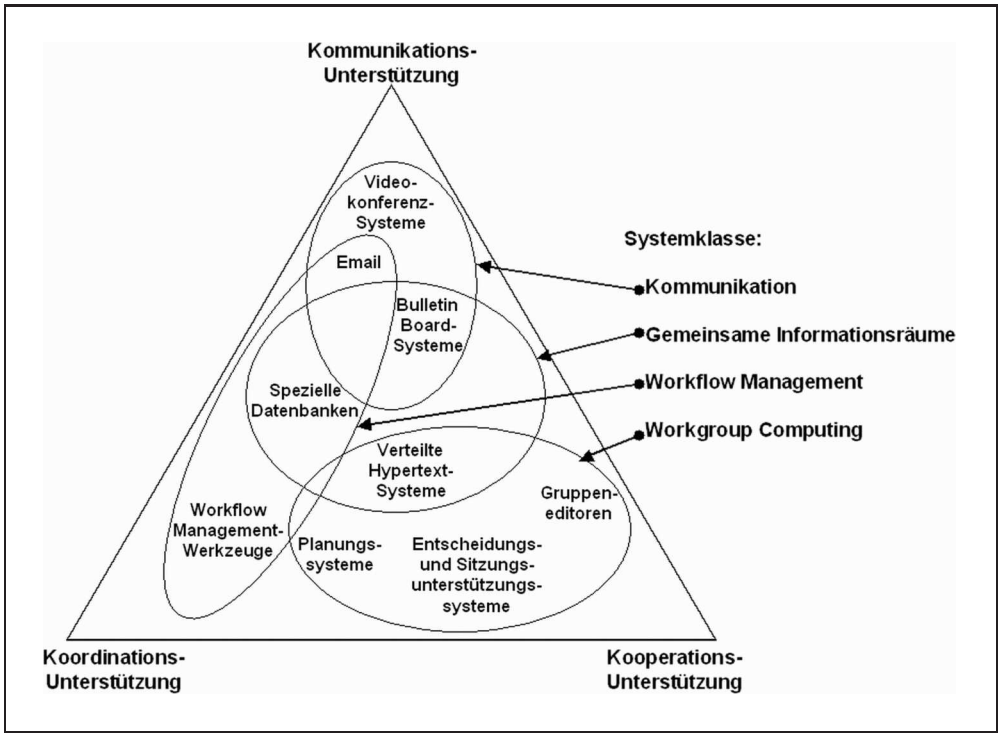

Abb. 4.5: Klassifikationsschema nach Unterstützungsfunktionen
(vgl. Teufel et al., 1995, S.27)

Teufel et al. (1995) gehen noch weiter und fassen die auf diese Weise positionierten CSCW-Systeme zu sogenannten Systemklassen zusammen:

1. Systemklasse *Kommunikation*:
   Die Systeme dieser Klasse ermöglichen primär die Kommunikation zwischen allen Beteiligten und überbrücken dabei räumliche und zeitliche Differenzen.

2. Systemklasse *Gemeinsame Informationsräume*:
   Mittels gemeinsamer Informationsräume können den Mitgliedern einer Gruppe Informationen über geeignete Zugriffsmechanismen zur Verfügung gestellt werden.

3. Systemklasse *Workflow Management*:
   Das Workflow Management beinhaltet sämtliche Aufgaben, die zur Modellierung, Simulation, Ausführung und Steuerung von in der Regel auf Unternehmensebene angesiedelten Abläufen notwendig sind. Die Werkzeuge des Workflow Management unterstützen die Beteiligten mittels geeigneter Software.

4. Systemklasse *Workgroup Computing*:
   Die Unterstützung schwach strukturierter Aufgaben auf Gruppenebene erfolgt mit den Werkzeugen dieser Systemklasse.

Weder die in dem vorigen Kapitel 4.6 genannten Groupware-Werkzeuge als Basiselemente, aus denen sich CSCW-Systeme zusammensetzen können, noch die in

diesem Kapitel (4.7) vollzogenen Klassifikationen sind umfassend oder gar trennscharf, geben aber die wichtigsten Elemente und deren grobe Zuordnung zu den grundlegenden Gruppenprozessen wieder. Auf die Benennung konkreter CSCW-Anwendungen wird zugunsten der Lesbarkeit dieser Arbeit verzichtet. Stattdessen wird an dieser Stelle auf weiterführende Literatur wie Gross & Koch (2007) oder Schwabe & Streitz (2001) verwiesen.

Kapitel 5

CSCL

5.1 Definitionen

Computerunterstütztes Lernen in der Gruppe ist mit dem Akronym CSCL (*Computer Supported Cooperative/Collaborative Learning*) assoziiert. Dahinter steht ein Forschungsgebiet, das sich ursprünglich aus der im Kapitel 4 behandelten Disziplin CSCW ableitet: *„CSCL kann als Anwendung von CSCW-Systemen im Bereich des Lehrens und Lernens verstanden werden; unter dieser Sichtweise werden CSCW-Systeme eben statt zum kooperativen Arbeiten zum kooperativen Lernen eingesetzt"* (Pfister & Wessner, 2001b, S.251).

McManus (1997) formuliert diese Sichtweise gleichsam plakativ wie provokant: *„CSCL = CSCW + CL"* (McManus, 1997, S.7) Nach ihr ergibt sich CSCL folglich aus der Fusion von computer-gestützter Gruppenarbeit (s. Kap.4) und kollaborativem Lernen (s. Kap.2.2.6), wobei der Schwerpunkt ihrer Betrachtungen auf der Computer-Unterstützung von Kooperationen (CSC) liegt und das kollaborative (oder kooperative) Lernen (L) als Anwendungsgebiet betrachtet wird. Der von CSCL erbrachte Output resultiert in einer Vielzahl von Computer-Systemen, die auf vielfältige Weise das Lernen in der Gruppe unterstützen sollen (vgl. McManus, 1997, S.8).

Dieser technisch-informatik-orientierten Sichtweise mag man nun folgen oder ihr widersprechen, jedoch wird diese allein dem Wesen (und den Inhalten) von CSCL nicht vollständig gerecht. In einer weiteren Aussage betonen Pfister & Wessner (2001b) einen weiteren Schwerpunkt: *„Computerunterstütztes kooperatives (oder kollaboratives) Lernen ist eine neue Entwicklung in einer langen Reihe von Versuchen, Lehren und Lernen durch den Einsatz von Computern und Computernetzen zu unterstützen und zu verbessern"* (Pfister & Wessner, 2001b, S.251). Entsprechend oben angeführter Argumentation steht hier das kollaborative Lernen (CL) als eine spezielle Form des Lernens (und Lehrens) im Mittelpunkt, welche mittels Computer-Unterstützung (CS) erfolgt und von eben dieser profitiert.

5.2 Forschungsdisziplinen

CSCL stellt also – wie auch CSCW – ein interdisziplinäres Forschungsgebiet dar, das von vier Disziplinen maßgeblich beeinflusst wird (vgl. Wessner, 2001, S.199ff; Haake et al., 2004, S.2):

- Die Psychologie behandelt grundlegende Aspekte und Fragestellungen des Lernens wie beispielsweise *„Wie lernen Menschen?"*.
- Die Soziologie befasst sich mit Problemen und Phänomänen des menschlichen Miteinanders, hier konkret mit Gruppenbildung, mit Kommunikation und Kooperation innerhalb von Gruppen sowie deren Förderung in verteilten Lerngruppen.
- Die Pädagogik versucht zu klären, welche Lehr- und Lernmethoden sich für diese spezielle Lernsituation (das computerunterstützte kooperative Lernen) eignen.
- Die Informatik behandelt die eher technische Seite des kooperativen Lernens und setzt sich einerseits mit der Frage auseinander, wie sich Informatiksysteme wirksam für kooperative Lernszenarien entwickeln lassen, und liefert andererseits geeignete Werkzeuge (s. Kap.4.6) zur Unterstützung des kooperativen Lernens.

Somit lässt sich CSCL im Schnittpunkt der genannten Disziplinen, namentlich der Informatik, der Soziologie, der Psychologie sowie der Pädagogik einordnen (s. Abb. 5.1).

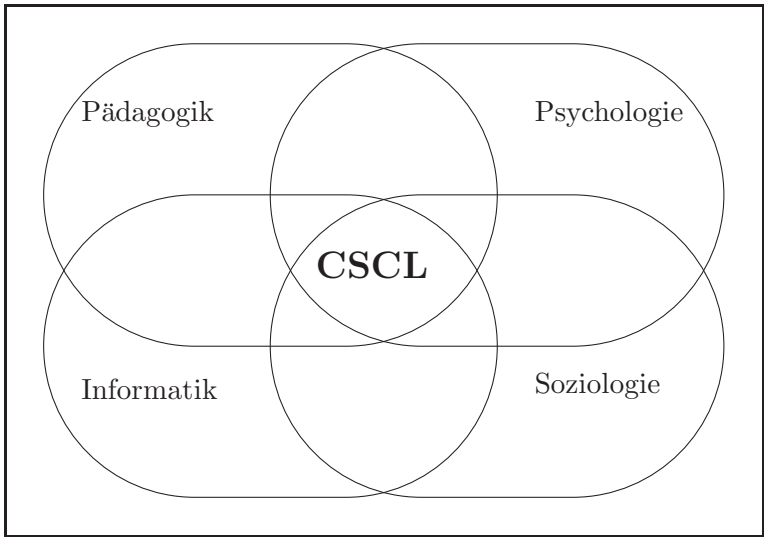

Abb. 5.1: Die Interdisziplinarität von CSCL

Ungeachtet des Fokus, mit dem man nun CSCL betrachtet, wird dieses mittlerweile – wie auch CSCW – als durchaus eigenständige Disziplin verstanden: *„Im*

engeren Sinne versteht man unter CSCL jedoch die Kombination von Computersystemen und pädagogisch-didaktischen Methoden, die die Vorteile kooperativen Lernens spezifisch realisieren. Unter dieser Sichtweise ist CSCL ein eigenständiges Forschungsfeld und eine eigene Lernform, auf die sich Befunde aus der sonstigen CSCW-Forschung und der traditionellen Pädagogik nur eingeschränkt übertragen lassen" (Pfister & Wessner, 2001b, S.251).

Unter diesem Gesichtspunkt lässt sich CSCL auch als spezifische Variante von E-Learning[1] begreifen: *„Elearning wird schließlich als Oberbegriff für alle Varianten internetbasierter Lehr- und Lernangebote verstanden"* (Kerres, 2001, S.14). So stand auch eine Vielzahl von Beiträgen der GI[2]-Jahrestagung *Informatik 2002* in Dortmund unter dem Motto *„CSCL - Kooperatives E-Learning"* (vgl. Schubert et al., 2002, S.221ff). Zweifelsohne hat CSCL also seinen Ursprung auch im E-Learning, daher wird diese Lernform nachfolgend charakterisiert,wodurch eine genauere Positionierung von CSCL ermöglicht wird.

5.3 E-Learning

5.3.1 Definitionen von E-Learning

Mit dem Begriff des E-Learning sind viele Definitionen verbunden. So schreibt beispielsweise Dittler (2003a): *„Was versteht man unter E-Learning? Der kleinste gemeinsame Nenner in Theorie und Praxis lautet: Integration von neuen Technologien in Aus- und Weiterbildung"* (Dittler, 2003a, S.76). Ähnlich formuliert es auch Mandl (2004): *„Grundsätzlich ist unter E-Learning das Lernen mit Hilfe elektronischer Medien zu verstehen. Die realisierten Methoden sind dabei sehr vielfältig. Sie reichen von Computer-Based-Training (CBT) oder Web-Based-Training (WBT) bis hin zu Online-Lernen"* (Mandl, 2004, S.17). Oder wie Bendel (2003) es ausdrückt: *„E-Learning ist Lernen, das mit Informations- und Kommunikationstechnologien (IKT) bzw. speziellen Lerntechnologien sowie mit Lernsystemen ermöglicht bzw. unterstützt wird"* (Bendel, 2003, S.17).

Mit der Erklärung von Bendel (2003) wird die definitorische Bestimmung zunächst und ohne Anspruch auf Vollständigkeit abgeschlossen, nicht jedoch ohne zuvor festzuhalten, dass E-Learning stets unter Einsatz elektronischer Medien stattfindet. E-Learning (und damit auch CSCL) ist jedoch nicht auf Technologien und Systeme begrenzt, sondern beinhaltet auch Aspekte aus der Didaktik, die stets zu berücksichtigen sind.

[1] Neben der in dieser Arbeit bevorzugten Schreibweise *E-Learning* wird dieses in der Literatur beispielsweise auch eLearning oder Elearning geschrieben (von engl. *electronic learning*; dt.: elektronisches Lernen).

[2] Gesellschaft für Informatik e.V.

5.3.2 Medientypen

Die Klassifizierung verwendbarer Medien, welche im Lernprozess als vermittelnder Faktor eine maßgebliche Rolle spielen, erfolgt anhand der auf Seiten von Sender und Empfänger benötigten Technik (vgl. LANG, 1978, S.17ff; DÖRING, 2003, S.40):

- **Primärmedien** erfordern keine besondere Technik seitens Sender oder Empfänger. Dazu zählen beispielsweise sämtliche Mittel zwischenmenschlicher Elementarkontakte.
- **Sekundärmedien** setzen einen Einsatz von Technik beim Sender voraus. Typische Beispiele sind Printmedien wie Bücher, Zeitschriften und Zeitungen.
- **Tertiärmedien** erfordern eine technischen Aufwand sowohl beim Sender als auch beim Empfänger wie beispielsweise Radio oder TV.
- **Quartärmedien** benötigen – wie auch Tertiärmedien – auf beiden Seiten (Sender und Empfänger) den Einsatz von Technik. Aufgrund von Besonderheiten der Technik ist aber die für die Massenkommunikation typische einseitige Sender-Empfänger-Beziehung aufgehoben (vgl. HERGET ET AL., 1999).

Die Quartärmedien kennzeichnen insbesondere die *Neuen Medien* (auch Multimedia), die – im Gegensatz zu den *alten Medien* wie dem Face-to-Face-Unterricht oder dem Buch (vgl. HAAKE ET AL., 2004, S.258) – als computerbasiert, multimedial, hyperstrukturiert, interaktiv, kommunikativ und multifunktional charakterisiert werden (vgl. LANG, 2002, S.29). Insbesondere wird ihnen ein großes Potential bei der Gestaltung von Lernprogrammen sowie bei der Modellierung und Unterstützung von Lernprozessen zugesprochen: *„Auf der Basis einer sich ständig verbessernden informations- und kommunikationstechnologischen Infrastruktur können so neuartige Lernprozesse realisiert werden, bei denen Lehrende und Lernende weder am gleichen Ort noch gleichzeitig mit dem Lernen beschäftigt sein müssen. ... Diese Technologien ermöglichen neue Lernformen (z.B. örtlich verteiltes zeitgleiches Gruppenlernen per Audiokonferenz oder zeitversetztes Gruppenlernen unter Nutzung von Diskussionsforen), neue Formen und Qualitäten des Lernmaterials (z.B. interaktive 3D-Simulationen, hypermediale Repräsentationen von Zusammenhängen) und neue Lernorte wie etwa die virtuelle Universität"* (WESSNER, 2001, S.197).

Speziell die Unabhängigkeit von Raum und Zeit machen diese neuen Medien auch interessant für die Unterstützung von Lernprozessen, bei denen der Lehrende zumindest partiell durch eine spezielle Software-Komponente[3] substituiert werden kann. Diese Eigenschaft ist auch für die vorliegende Arbeit relevant, da das VitaminL-System als Forschungswerkzeug dieser Arbeit auf der Entkopplung der Lerngruppen von Raum und Zeit aufsetzt.

[3] Solch eine Komponente stellt einen sogenannten *virtuellen Tutor* dar. Dieses Konzept wird im Kapitel 6 behandelt.

Zwecks eines besseren Verständnisses von E-Learning – und damit auch von CSCL – wird E-Learning nun eingeordnet in die Lernlandschaft und gegen ähnliche Begriff abgegrenzt, die oftmals und fälschlicherweise mit E-Learning gleichgesetzt werden.

5.3.3 Positionierung des E-Learning

Lang (2002) unterscheidet die drei grundsätzlichen Lern-Domänen – Präsenzlernen, Distanzlernen und Online-Lernen – anhand der ihnen typischerweise zuzuordnenden Medientypen sowie anhand der Distanz zwischen Lehrenden und Lernenden und setzt das E-Learning in Beziehung zu diesen Lern-Domänen.

5.3.3.1 Präsenzlernen

Die traditionelle Lernform basiert typischerweise auf dem Einsatz von Primär- und Sekundärmedien, bei dem Lehrende und Lernende sich real und unmittelbar treffen, so dass Wissensvermittlung als face-to-face-Lernen stattfinden kann (bspw. in Form von Frontalunterricht). Sekundärmedien (wie Bücher, Schaubilder etc.) werden unterstützend eingesetzt. Insgesamt zeichnet sich die Lernsituation im Präsenzlernen durch ein hohes Maß an Interaktionen und Kommunikation zwischen Lehrenden und Lernenden aus (vgl. Lang, 2002, S.34f).

5.3.3.2 Distanzlernen

Distanzlernen basiert auf der Unabhängigkeit von Raum und Zeit und wird darüber hinaus unterstützt durch die allgemeine Verfügbarkeit einer Vielzahl medialer Informationsspeicher (Skripte, Bücher, sonstige Dokumente). Infolgedessen können alle Elemente des sogenannten *didaktischen Dreiecks* (bestehend aus Lehrer, Lerner und Lernmaterialien; vgl. Klauss, 2005, S.26f) räumlich und zeitlich entkoppelt miteinander interagieren, wobei der Grad an Interaktivität und Kommunikation zwischen den Beteiligten wesentlich geringer ausfällt als beim Präsenzlernen. Diese spezielle Lernform erfordert daher vom Lernenden hohe Eigendisziplin, Selbstorganisation und gut strukturiertes Zeitmanagement.

5.3.3.3 Online-Lernen

Der Einsatz von Neuen oder Quartärmedien prägt das Online-Lernen als eigenständige Lernform, welches üblicherweise auf der computer-basierten Vernetzung der Beteiligten basiert: *„Beim Online-Lernen (Lernen im Netz) greifen die Lernenden und Tutoren auf einen Server zu, auf dem die relevanten Daten gespeichert sind. Tutoren und Lernende können untereinander synchron oder asynchron kommunizieren"* (Winkler & Mandl, 2003, S.192). Online-Lernen besitzt gegenüber den zuvor genannten Lernformen potentielle Vorteile: Neben

der Unabhängigkeit von Ort und Zeit, ermöglicht durch die Nutzung computervermittelter Kommunikation (s. Kap.3.1), bietet eben diese Computerunterstützung dem Lernenden Verknüpfungen zu weiteren Lernressourcen wie beispielsweise Foren für Gruppendiskussionen. Ein weiterer Vorteil liegt in der schnellen Verfügbarkeit von Teilnehmern und Medien, der durch Einsatz von virtuellen Tutoren noch erhöht werden kann. Insbesondere zeichnet sich auch diese Lernform durch hohe Interaktivität und Kommunikation aus.

In diesem Gesamtzusammenhang lässt sich E-Learning also als spezielle Variante des Online-Lernens begreifen, das sich vornehmlich auf den Einsatz neuer Medien stützt (vgl. Kerres, 2001, S.14; Mandl, 2004, S.17), ohne jedoch auf die ausschließliche Nutzung von Computern und Computernetzwerken begrenzt zu sein. Tatsächlich finden auch Video- und Audio-Kassetten/-Bänder sowie DVDs, CDs und CD-ROMs im E-Learning Verwendung. E-Learning muss also nicht zwingend online stattfinden, wenngleich dieser Anteil in Folge technologischer Entwicklungen – namentlich des Ausbaus für jederman verfügbarer und bezahlbarer Breitbandanschlüsse wie DSL – zugunsten von computer-gestütztem, online-basiertem E-Learning abnehmen wird. Die Positionierung von E-Learning bezüglich der drei zuvor skizzierten Lern-Domänen kann der Abbildung 5.2 entnommen werden.

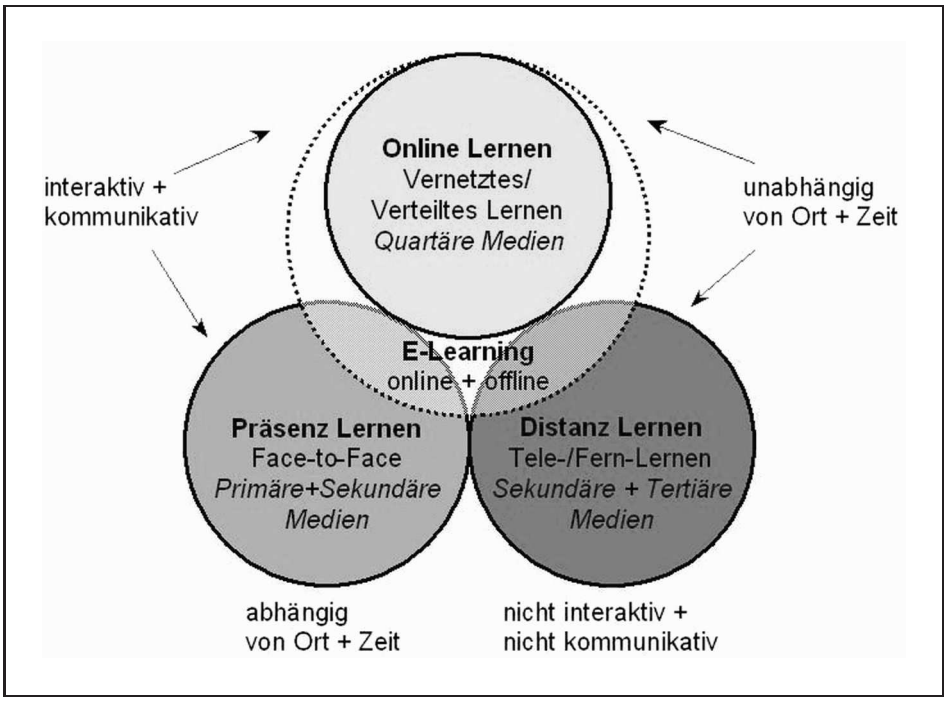

Abb. 5.2: Positionierung des E-Learning
(vgl. Lang, 2002, S.35ff)

5.3.4 Technische Formen des E-Learning

E-Learning als Erweiterung des Online-Lernen bedient sich vieler unterschiedlicher Medientypen und technischer Ansätze, da diese Arbeit jedoch schwerpunktmäßig

im Bereich CSCL anzusiedeln ist, werden nachfolgend ausschließlich diejenigen Formen des E-Learning betrachtet, die unter überwiegender Nutzung von Computern stattfinden.

5.3.4.1 Computer-Based-Training

Computer-Based-Training (CBT) kann als eine der ältesten und einfachsten Formen von E-Learning verstanden werden: *„Computer-Based-Training (CBT) bezeichnet Lernprogramme, die seit den 80er Jahren auf der Basis von Computern zum Selbstlernen eingesetzt werden"* (Winkler & Mandl, 2003, S.192). Es handelt sich bei CBTs um eigenständige Applikationen, die – im Gegensatz zu anderen E-Learning-Formen – ohne Anbindung an ein Netzwerk (resp. das Internet) auskommen. CBTs werden dazu auf Medien wie Disketten, CD-ROMs oder DVDs ausgeliefert und lokal auf einem Computer installiert. Nach dem Programmstart erfolgt in der Regel eine kurze Einführung in das Programm und seine Bedienung, bevor die eigentliche Vermittlung von Wissen beginnt. Die behandelten Themen werden vom Programm entweder in einer vorgegeben Reihenfolge abgearbeitet oder aber dem Benutzer zur freien Auswahl in einer Übersichtsseite (engl. *site map*) angeboten, so dass der direkte Zugriff auf einzelne Themen ermöglicht wird (vgl. Dittler, 2003b, S.25).

Die Abarbeitung eines einzelnen Themas folgt dabei meist einem klassischen Ablauf, beginnend mit der Präsentation der Inhalte eines Lerngebiets durch das Programm, gefolgt vom Bearbeiten von Übungen zum soeben erworbenen Wissen durch den Lernenden. Den Abschluss bildet im Idealfall ein differenziertes Feedback durch das CBT, wodurch der Benutzer eine Erfolgskontrolle erhält mit der Möglichkeit, noch vorhandene Defizite aufzuarbeiten (vgl. Dittler, 2003b, S.25).

CBT richtet sich – im Gegensatz zu anderen E-Learning-Formen – prinzipiell an den individuellen Lernenden. Das Lernen findet verteilt und asynchron statt, also wie im E-Learning üblich entkoppelt von Ort und Zeit. Es handelt sich ferner um ein medienzentriertes, individuelles Lernen, das durch einen hohen Grad an Interaktivität zwischen Lernenden und CBT geprägt ist (vgl. Dittler, 2003b, S.12,S.26f).

5.3.4.2 Web-Based-Training

Mit zunehmender Vernetzung privater wie geschäftlicher Rechner durch Anbindung an beispielsweise das Internet gewinnen *Web-Based-Trainings* (WBTs) immer mehr an Bedeutung: *„Der Begriff* Web-Based-Training *(WBT) beschreibt ein Lernen über Netzumgebungen wie Internet, Intranet oder Extranet. Die Grundlemente sind Informationssysteme (z.B. Datenbanken) und Lernprogramme mit Übungen, Tests etc. Somit können WBTs als technische Weiterentwicklung von CBTs betrachtet werden"* (Mandl, 2004, S.17). Unabdingbare Voraussetzung ist jedoch die Verfügbarkeit eines Internetzugangs (vgl. Dittler, 2003b, S.153).

Aufgrund der Nutzung eines Netzwerks ermöglichen WBTs – im Gegensatz zu CBTs – nicht nur individuelles Lernen, sondern auch das kooperative Lernen in Gruppen: *„Im Unterschied zu CBT ist ein kooperatives Lernen zweier oder mehrerer Teilnehmer gleichzeitig mit einem Programm möglich. Lernende können in einem WBT beispielsweise gemeinsam ein Problem oder eine Aufgabenstellung bearbeiten und dabei nicht nur vom Computer, sondern auch voneinander lernen. Der Computer übernimmt damit beim Einsatz von WBT nur die Rollen eines Vermittlers zwischen zwei Lernenden"* (Dittler, 2003b, S.12). WBTs lassen sich folglich so gestalten, dass die Vermittlung von Wissen nicht ausschließlich durch das WBT stattfindet, sondern vorrangig in der Kooperation der Lernpartner vollzogen wird (vgl. Dittler, 2003b, S.154).

Neben diesem Aspekt des kollaborativen Lernens sind WBTs auch deutlich schneller und einfacher aktuell zu halten, da nahezu sämtliche relevanten Daten (Lernmaterialien wie Programmcode) zentral auf einem Server vorrätig gehalten und gepflegt werden können: *„Im Internet verbleiben Web-Based-Trainings in der Regel unter dem unmittelbaren Zugriff des Herstellers und können daher in kürzeren Abständen angepasst und aktualisiert werden. Zudem ist der Distributionsweg vom Anbieter bis zum Endkunden deutlich kürzer"* (Dittler, 2003b, S.273).

Somit stellen WBTs eine Lernform dar, die – wie auch CBTs – verteiltes Lernen ermöglichen, das als medienzentriertes Lernen mit einem hohen Grad an Interaktivität stattfindet. Es kann asynchron stattfinden, aber in solchen Fällen, wo auch kollaboratives Lernen unterstützt wird, auch synchron.

5.3.4.3 Lernplattformen

Lernplattformen oder *E-Learning-Portale* lassen sich als Weiterentwicklung der WBTs auffassen: *„Unter einer E-Learning-Plattform versteht man ein System, das es ermöglicht, innerhalb eines Unternehmens ein virtuelles Bildungszentrum aufzubauen. Kern der Plattform ist die Verwaltung von E-Learning-Angeboten und jeglicher Art von Lernmedien sowie der Anwenderdaten"* (Mandl, 2004, S.18). Die Anwendung bereits vorgestellter Technologien (s. Kap.3.1 und Kap.4.6) ermöglicht die Integration weiterer, externer Informationsquellen (bspw. in Form von digitalen Bibliotheken) sowie die Bereitstellung von Arbeitsbereichen sowohl für den einzelnen Lerner als auch für Lerngruppen (vgl. Schulmeister, 2001, S.165; Winkler & Mandl, 2003, S.192; Döring, 2003, S.115).

Der Mehrwert von Lernplattformen gegenüber WBTs ergibt sich nicht nur aus der in der Regel umfangreicheren Sammlung von Lernmaterialien und die Integration zusätzlicher Dienste, sondern auch durch die Form und Intensität der Betreuuung von Lernenden. Lernplattformen enthalten üblicherweise umfangreiche Coaching-Funktionen: *„Ein oder mehrere Fachexperten oder Bildungsbegleiter stehen den Lernenden für technische, aber auch vor allem für fachliche und inhaltliche Fragen zur Verfügung. Individuelle Empfehlungen können per Mail oder Chat (synchrone und asynchrone Kommunikation) ebenso gemeinsam erarbeitet werden, wie Sachfragen im direkten Austausch besprochen und geklärt werden können"* (Dittler, 2003b,

S.12). Folglich ergibt sich aus dem Einsatz von Lernplattformen – im Vergleich zu traditionellen Lehrveranstaltungen mit hohem Präsenzanteil und *face-to-face*-Kommunikation – ein höherer Bedarf an qualifizierten Betreuern bei gleichzeitig höherer Reichweite.

5.3.4.4 Virtuelle Seminare

Virtuelle oder Online-Seminare können als virtualisierte Gegenstücke zu traditionellen Seminaren, Workshops und Vorlesungen angesehen werden: *„Online-Seminare bieten virtuelles Lernen in einer speziellen Organisationsform, die in der Regel an Präsenzseminaren orientiert ist"* (Schulmeister, 2001, S.255). Dieses überwiegend synchron stattfindende Online-Lernen ergibt sich aus der Kombination von mediengestütztem Lernen und Präsenzveranstaltung mit dem Ziel, die Vorteile beider Formen nutzen zu können (vgl. Dittler, 2003b, S.203):

- Die Möglichkeit der direkten Kommunikation zwischen allen Teilnehmern – realisiert durch entsprechende Dienste und Applikationen wie beispielsweise Videokonferenzen – bietet einerseits Lernenden die Möglichkeit, zeitnah und spontan Rückfragen zu aktuellen Inhalten zu stellen, und gibt andererseits Dozenten die Gelegenheit, durch vertiefende Fragestellungen an das Auditorium zum besseren Verständnis der Inhalte beizutragen.

- Die multimediale Aufbereitung von Inhalten ermöglicht die Illustration komplizierter Sachverhalte und komplexer Zusammenhänge. Ferner können elektronische Fragebögen und andere Verfahren eingesetzt werden, um im Anschluss an die Vermittlung von Lerninhalten deren Überprüfung (inkl. Rückmeldung an den Lernenden) effizient durchzuführen.

- Da räumliche Beschränkungen weitgehend entfallen, lassen sich mit virtuellen Seminaren – wie auch mit CBTs und WBTs — größere Lerngruppen erreichen als in traditionellen Veranstaltungen.

Nachteilig sind der erhöhte Arbeitsaufwand aller Beteiligten sowie die Notwendigkeit einer geeigneten Moderation zwecks Stimulation und Motivation der Lernenden zu nennen (vgl. Schulmeister, 2001, S.255). Auch der erhöhte technische und organisatorische Aufwand darf nicht vernachlässigt werden (vgl. Kerres, 2001, S.299).

Der typische Ablauf eines virtuellen Seminars beginnt im Vorfeld mit der Vorbereitung des Lerninhalts und dem Zusammenstellen aller (digitaler) Materialien durch den Dozenten. Die Lernenden melden sich zum Beginn der Veranstaltung mit ihrem Computer bzw. darauf vorhandener Software (in der Regel genügt ein Internet-Browser) an der zugehörigen Internet-Adresse an und nehmen somit an ihrem Rechner sitzend am virtuellen Seminar teil. Die Lerninhalte werden über das Internet verschickt und gleichzeitig auf die PCs aller angemeldeten Teilnehmer übertragen. Dies kann in Form einer Videokonferenz erfolgen oder durch Senden

von digitalen Präsentationen; in der Regel werden verschiedene Medien miteinander kombiniert. Mittels Chat oder ähnlicher Dienste besteht die Möglichkeit synchroner Kommunikation zwischen allen Teilnehmern. Hierbei ist jedoch zu beachten, dass dies – wie auch in traditionellen Veranstaltungen – nur bis zu einer bestimmten Teilnehmerzahl sinnvoll möglich ist.

Verschiedene Konzepte virtueller Seminare werden in (Schulmeister, 2001, S.266ff) skizziert. Kerres (2001) spricht in diesem Zusammenhang auch vom *virtuellen Klassenzimmer* (Kerres, 2001, S.299). Prinzipiell lassen sich virtuelle Seminare als eine Form des verteilten Lernens begreifen, die aber ähnlich wie traditionell durchgeführte Vorlesungen und Seminare eher personenorientiert stattfindet und weniger medienzentriert wie beispielsweise CBTs und WBTs. Obwohl es sich im Prinzip um eine synchrone Form des E-Learning handelt, die sich – bis zu einer gewissen Teilnehmerzahl – auch durch ein hohes Maß an möglicher Interaktion auszeichnet, können virtuelle Seminare auch auf Datenträgern gespeichert und bei Bedarf genutzt werden. Diese *on demand*-Seminare büßen dann allerdings aufgrund der Asynchronität auch ihren Anteil an Interaktion ein (vgl. Dittler, 2003b, S.204f).

5.3.4.5 Virtuelle Universitäten

Die konsequente Fortführung der bisher genannten Ansätze resultiert in der Umsetzung von virtuellen Universitäten, in denen ganze Curricula (inkl. Prüfungen) in virtueller Form stattfinden. Nach Schulmeister (2001) vollzieht sich die Etablierung virtueller Universitäten in vier großen Entwicklungslinien (vgl. Schulmeister, 2001, S.51):

- *„Fernuniversitäten werden virtuell"*:
  Die existierenden Fernuniversitäten erweitern ihr Angebot um virtuelle Kurse oder komplett virtuell durchgeführte Studiengänge (vgl. Schulmeister, 2001, S.53ff).

- *„Die Alma Mater virtualisiert sich"*:
  Klassische Präsenzuniversitäten legen sich mit virtuellen Veranstaltungen einen zweiten Distributionsweg respektive ein zweites Standbein zu. Zu dieser Entwicklunglinie zählen auch virtuelle Kooperationen wie beispielsweise der virtuelle Campus der Universitäten Hannover-Hildesheim-Osnabrück (vgl. Schulmeister, 2001, S.62ff).

- *„Neugründung virtueller Universitäten"*:
  Rein virtuelle Organisationen werden gegründet mit dem Ziel, virtuelle Ausbildungsangebote zu vermarkten, zu verwalten und zu organisieren. Oftmals werden diese Angebote gar nicht von diesen Institutionen selbst entwickelt, veranstaltet und betreut, sie greifen vielmehr auf bereits vorhandene Angebote anderer Bildungseinrichtungen zu und fungieren somit als eine Art *Bildungsmakler* (vgl. Schulmeister, 2001, S.93ff).

- *„Corporate Universities"*:
  Große Firmen versuchen, den ständigen Fort- und Weiterbildungsbedarf ihrer eigenen Mitarbeiter zu decken und werden durch die Gründung entsprechender virtueller Bildungseinrichtungen selbst zum Bildungsanbieter (vgl. Schulmeister, 2001, S.115ff).

Die virtuelle Universität als solche ist nach Expertenmeinung unvermeidbar (vgl. Schulmeister, 2001, S.121). Bis zu ihrer Etablierung sind jedoch noch wichtige Fragen zu klären und offene Punkte zu diskutieren. So ist insbesondere die Nachfrage in der Zielgruppe der Studienanfänger noch ungeklärt, da bisherige Studien ergeben haben, dass gerade die 18- bis 24-jährigen Studierenden ein hohes Kontaktbedürfnis aufweisen, dessen Befriedigung im Alltag einer virtuellen Universitäten mit überwiegend computer-vermittelten Kontakten kaum stattfinden wird (vgl. Schulmeister, 2001, S.120).

5.3.5 Einordnung von CSCL

Wollte man CSCL nun innerhalb des E-Learning und der genannten Lern-Domänen positionieren, so handelt es sich nach Einschätzung des Autors unbedingt um eine spezielle Variante des Online-Lernens. Je nach Ausprägung des verwendeten Systems können die Mitglieder einer Lerngruppe in einer konkreten Lernsituation sich an einem Ort befinden oder über mehrere Orte verteilt am Lernprozess teilnehmen. Folglich rückt CSCL auch in die Nähe des Distanzlernens, so dass sich eine Positionierung von CSCL innerhalb der Lern-Domänen ergibt, die in der nachfolgenden Abbildung 5.3 wiedergegeben ist.

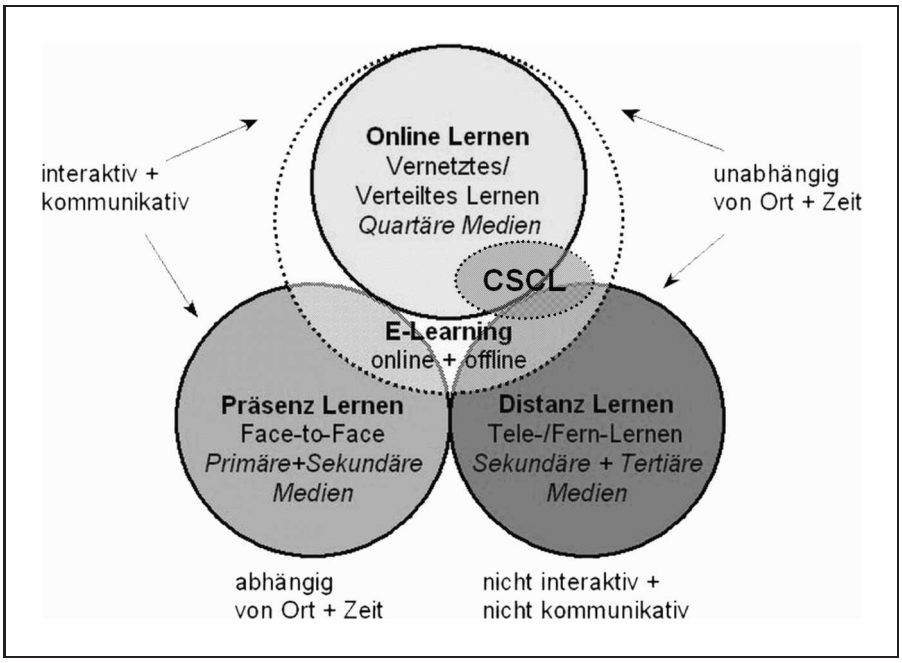

Abb. 5.3: Einordnung von CSCL nach Lern-Domänen

Dieser Unterschied ermöglicht nicht nur eine klare Abgrenzung zwischen CSCL und E-Learning, sondern verdeutlicht auch einen klaren Vorteil von solchen computergestützten Lernformen, die auf Kooperation basieren, gegenüber eher traditionellen E-Learningformen wie CBTs und WBTs: hier kann nicht nur der bereits erwähnte positive Einfluss der neuen Medien greifen und sich entfalten (s. Kap.5.3.2), auch der aktive Diskurs zwischen den Teilnehmern spielt eine wichtige Rolle im Lernprozess. Dabei geht es nicht ausschließlich um den Austausch von verteilt vorhandenem Wissen und damit verbunden dem Aufbau einer gemeinsamen Wissensbasis: *„Bei der gemeinsamen Bearbeitung von Aufgaben oder eines Problems und in der Kommunikation mit den anderen wird die eigene Position realistischer eingeschätzt und das eigene Selbstbild leichter herausgefunden. In der Gruppe wird der Einzelne nicht nur stärker herausgefordert, sondern findet gleichzeitig auch Bekräftigung und Unterstützung"* (Grune & de Witt, 2004, S.28).

5.4 Didaktische Konzepte

CSCL als Variante des E-Learning hat aber nicht nur eine technologie-getriebene Sichtweise, sondern wird als interdisziplinäres Forschungsgebiet auch von der Pädagogik, respektive der Didaktik geprägt. Zum besseren Gesamtverständnis von CSCL werden nachfolgend wesentliche Elemente und Konzepte aus diesem Bereich skizziert.

5.4.1 Der Konstruktivismus

Der Konstruktivismus steht in enger Beziehung zum sozialen Pragmatismus nach Dewey und stellt die lerntheoretische Grundlage für (computerunterstütztes) kooperatives Lernen dar: *„Der lerntheoretische Konstruktivismus beschreibt den Lernprozess als aktiven Konstruktionsprozess des Lernenden. Auf der Basis der eigenen Erfahrungen gestaltet der Lernende seinen Lern- und Verständnisprozess. Der Lehrende begleitet diesen Prozess durch individuell dosierte Unterstützung. Wissen kann demnach nicht wie ein feststehendes Produkt vom Lehrenden zum Lernenden vermittelt werden, sondern wird vielmehr vom Lernenden vor dem Hintergrund dessen Erfahrungen aktiv konstruiert"* (Hohenstein & Sander, 2005, S.8). Wissen wird also stets generiert durch Wechselwirkung zwischen dem Lernenden und seiner Umwelt (vgl. Kriz & Nöbauer, 2002, S.76f).

Insgesamt ist Lernen aus konstruktivistischer Sicht ein aktiver, selbstgesteuerter, konstruktiver, situativer, sozialer Prozess (vgl. Klauser, 2002, S.5; Martens, 2003, S.127; Mandl, 2004, S.19; Grune & de Witt, 2004, S.38):

- *Lernen als aktiver Prozess*:
  Erst die aktive Auseinandersetzung von Lernenden mit den Lerninhalten ermöglicht die individuelle Konstruktion von Wissen: *„Nach dem konstruktivistischen Paradigma ist unser Wissen über die Welt keine passive Abbildung*

objektiver Sachverhalte, sondern das Ergebnis eines mentalen Konstruktionsprozesses" (Wessner, 2001, S.196).

- *Lernen als selbstgesteuerter Prozess*:
  Der Lernende selbst zeichnet sich verantwortlich für die Organisation, Gestaltung, Durchführung und Kontrolle seines Lernprozesses.
- *Lernen als konstruktiver Prozess*:
  Die individuelle Konstruktion von Wissen auf der Basis von Erfahrungen und Vorwissen bildet den Kerngedanken des Konstruktivismus.
- *Lernen als situativer Prozess*:
  Erst der Bezug zu einem relevanten Kontext oder Lerngegenstand sichert dem Lernprozess eine gewisse Dauerhaftigkeit und Nachhaltigkeit zu.
- *Lernen als sozialer Prozess*:
  Lernen findet in der Regel unter Einbeziehung sozialer Komponenten statt.

Klauser (2002) veranschaulicht die konstruktivistisch geprägten Position wie in Abbildung 5.4.

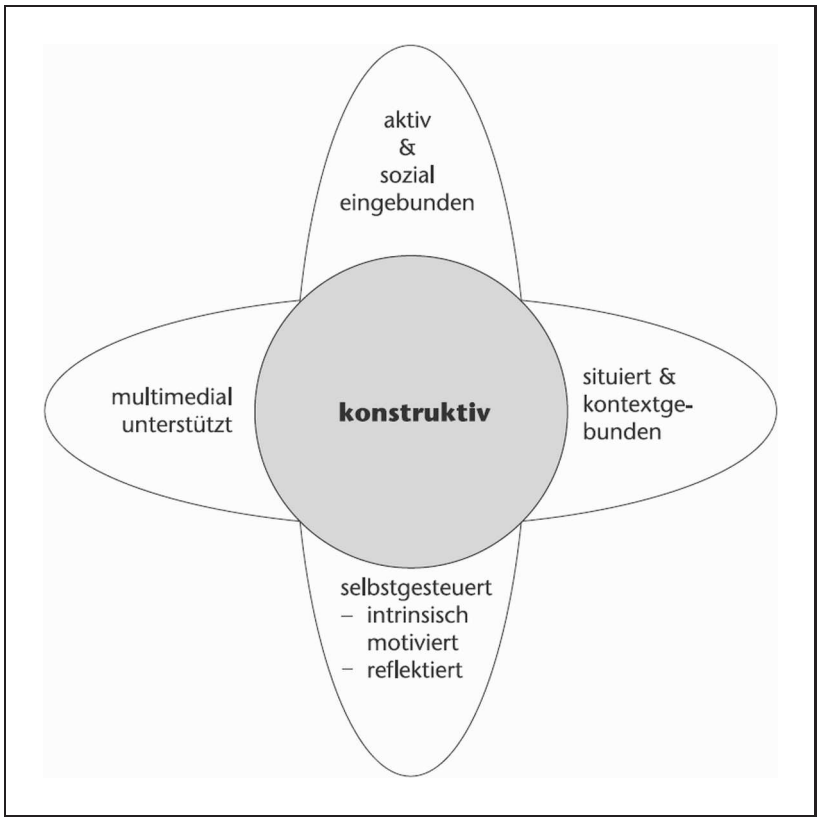

Abb. 5.4: Charakteristika des Lernprozesses aus konstruktivistischer Sicht (Klauser, 2002, S.5)

Der Einsatz neuer Medien (sprich: Multimedia; s. Kap.5.3.2) kann diese konstruktivistisch-orientierten Lernprozesse fördern: Das Lernen unter Zuhilfenahme multimedialer Programme stellt allein aufgrund der Interaktionen mit dem

Programm bereits einen aktiven Prozess dar, der zudem als selbstgesteuerter Prozess aufgefasst werden kann. Durch Hinzunahme weiterer Beteiligter und der daraus resultierenden Bildung von Kooperationen kann die Interaktion noch intensiviert werden. Wenn die inhaltliche Gestaltung des Lernprogramms so erfolgt, dass ein authentischer Kontext geschaffen wird, ist auch das Kriterium der Situiertheit erfüllt (vgl. Martens, 2003, S.127; Wessner, 2001, S.197).

5.4.2 Kollaboratives Lernen

Lernen ist aus konstruktivistischer Sicht auch immer ein sozialer Prozess, an dem folglich mehrere Lernende beteiligt sind. McManus (1997) definiert das kollaborative Lernen folgendermaßen: *„Collaborative Learning (CL) is defined as students working together in small, heterogenous groups to achieve a common academic goal, such as completion of a worksheet, an assignment or a project"* (McManus, 1997, S.7).

Wichtig ist nicht nur die Teilnahme mehrerer Lernender, sondern vor allem deren *aktive* Teilnahme an der Lernsituation: *„Beim kollaborativen Lernen wird die Verantwortung für die verschiedenen Aufgaben des Moderators auf alle Kursteilnehmer ausgedehnt. Die Teilnehmer werden in das Kursgeschehen eingebunden, was ihre Motivation und ihr Engagement meist deutlich erhöht"* (Hansen, 2003, S.1). Ferner findet gerade in einem aktiv geführten Diskurs eine bewusste Externalisierung vorhandenen Induvidualwissens statt, die überhaupt erst das Explizitmachen dieses Wissens ermöglicht. Dies und die in einem CSCL-System technisch durchführbare Speicherung solchen expliziten Wissens ermöglicht dessen Austausch zwischen den Beteiligten (auch zu einem späteren Zeitpunkt) und damit auch die Nachhaltigkeit bei jedem einzelnen Teilnehmer (vgl. Soller, 2001, S.58; Pfister & Wessner, 2001b, S.256).

Im Rahmen von CSCL findet ein kollaboratives Lernszenario stets computervermittelt statt, wobei verschiedene Varianten existieren. Pfister & Wessner (2001b) nennen in diesem Zusammenhang die folgenden typischen Szenarien (vgl. Pfister & Wessner, 2001b, S.255):

- Lokale Gruppen bearbeiten in einem Raum gemeinsam Lehrmaterial. Diskussion von Inhalten findet face-to-face statt, der Computer wird lediglich als unterstützendes Werkzeug während der Arbeit eingesetzt.
- Verteilte große Gruppen diskutieren ihre Lerninhalte asynchron über einen längeren Zeitraum. Von diesem Vorgehen machen beispielsweise netzbasierte Weiterbildungskurse, die von einem Tutor moderiert werden, Gebrauch.
- Verteilte kleine Gruppen tauschen selbstorganisiert und synchron Wissen aus, bearbeiten ebenso ihre Lernmaterialien und diskutieren.

In den beiden letztgenannten Szenarien wird der Computer nicht nur als reines Werkzeug zur Bearbeitung von Lernmaterialien verwendet, er rückt unaufdring-

lich in den Mittelpunkt des Geschehens und ist gleichsam Medium wie Transportmittel für die darzustellenden bzw. zu bearbeitenden Inhalte. Dennoch ist ein CSCL-System nicht ausschließlich als reines technisches System zu betrachten, das Lerninhalte präsentiert und die Kommunikation zwischen den beteiligten Lernenden ermöglicht, vielmehr wird auch stets eine pädagogisch-didaktische Komponente benötigt, die den Lernprozess unterstützt und fördert (vgl. Pfister & Wessner, 2001b, S.251). Einen verbreiteten Ansatz bilden die sogenannten *Intelligenten Tutoriellen Systeme* (ITS), die in Kapitel 6 nähere Betrachtung erfahren sollen.

5.4.3 Situiertes Lernen

Eines der Grundelemente des Konstruktivismus ist das situierte Lernen, demzufolge Lernen immer eingebettet in einen sozialen Kontext stattfindet (vgl. Kriz & Nöbauer, 2002, S.77): *„So geht zunächst der Ansatz des* situierten Lernens *von zwei Prämissen aus: Einbettung der Lernumgebung/Lernziele in einen authentischen Kontext und Anregung/Förderung sozialer Interaktion und Kooperation in Lernsituationen"* (Grune & de Witt, 2004, S.39). Dies ist auch bei der Gestaltung von Lernumgebungen bzw. Lerninhalten zu beachten: *„Für den Lernprozess bedeutet dies, dass das Augenmerk darauf gelegt werden muss, dass Kognition in situ geschieht, kontextuell gebunden oder »situiert« ist"* (Schulmeister, 1997, S.75).

Ein Anwendungsfall situierten Lernens ist der Ansatz des verankerten Lernens (engl.: *anchored instruction*): *„Beim Anchored Instruction-Ansatz geht es darum, wie vernetztes Wissen und Können mit Hilfe von narrativen videobasierten Fallpräsentationen, die als kognitiver, motivationaler und emotionaler Anker bei der Problembearbeitung dienen, erworben, angewandt und gefestigt werden kann. Das narrative Format dient dabei der situierten Einbettung der Problemstellung"* (Klauser, 2002, S.7).

Es wird also versucht, mit Hilfe von Videos als Anker eine Lernsituation zu schaffen, anhand derer den Lernenden ein Ansatz zur Identifikation mit der zu bearbeitenden Aufgabe an die Hand gegeben wird (vgl. Grune & de Witt, 2004, S.39). Somit verdeutlicht das verankerte Lernen in anschaulicher Form den sinnvollen Einsatz multimedialer Elemente in Lernumgebungen zur Unterstützung und Förderung des Lernprozesses. Wichtiger als der Einsatz der neuen Medien ist jedoch hierbei der Versuch, den individuellen Erfahrungsschatz eines Lernenden zu berücksichtigen und durch Verknüpfung mit authentischen Szenarien die Konstruktion neues Wissens anzuregen (vgl. Macha, 2001, S.2). Wo der Einsatz von Videomaterial nicht machbar ist, muss dies auf andere Weise erfolgen, beispielsweise durch Graphiken oder in Form von textuellen Beschreibungen relevanter Problemsituationen.

5.4.4 Problembasiertes Lernen

Das problembasierte Lernen (engl.: *problem-based learning*) ist ein mittlerweile etabliertes Lernkonzept, dessen Ursprung in den 1960er Jahren liegt. Dieser Ansatz

wurde seinerzeit erstmalig in der Ausbildung von Medizinern angewendet mit den Zielen, eine interdisziplinäre Ausbildung durchzuführen, in deren Verlauf der Lernende von Anfang an aktiv und kooperativ das eigenständige Problemlösen erlernt und während dieses Lernprozesses ein intensives Anwendungstraining von Wissen und Können erfährt. Inzwischen kommt dieser Ansatz in zahlreichen Bereichen wie beispielsweise der Hochschulausbildung, der medizinischen Ausbildung und der Ausbildung von Lehrern erfolgreich zum Einsatz (vgl. Klauser, 2002, S.3f; Hoffmann, 2004, S.245f).

Der typische Verlauf eines problembasierten Lernszenarios beginnt mit der (idealerweise multimedialen, computer-gestützen) Präsentation des zu bearbeitenden Problems und einer anschließenden Diskussion über anwendbare Lösungsansätze und -strategien, wobei auch mögliche Defizite zu berücksichtigen und zu klären sind, so dass als Ergebnis eine konkrete Problemlösungsstrategie festgelegt werden kann. Im Zuge der Durchführung der vereinbarten Strategie sollte eine ständige Überprüfung der erzielten Ergebnisse stattfinden, so dass gegebenenfalls eine Korrektur bzw. ein Rücksprung zu vorigen Phasen erfolgen kann. Am Ende stehen Sichtung und Bewertung der Ergebnisse sowie eine Evaluation des Gesamtprozesses (vgl. Hoffmann, 2004, S.247ff).

Ähnlich wie das situierte Lernen verfolgt also auch der Ansatz des problembasierten Lernens die Idee, dass eine praxisnahe Lernsituation wesentlicher Bestandteil erfolgreichen Lernens darstellt. Das problembasierte Lernen geht jedoch noch einen Schritt weiter und stellt das der Lernsituation innewohnende Problem und dessen Bearbeitung in den Mittelpunkt der Betrachtungen und damit des Lernprozesses, wohingegen die Präsentation von Wissen, Information und Inhalt *nur* eine unterstützende Funktion besitzen. Klauss (2005) illustriert dies anschaulich am Beispiel einer Theateraufführung: *„While the problem itself plays the main role in this theatralical production about learning, the presentation of knowledge, information and content play supporting roles. They are also very important and indispensable to the success of the performance, but the main part belongs to the learner itself"* (Klauss, 2005, S.37). Wichtig sowohl für dieses Beispiel wie für das problembasierte Lernen allgemein sind nicht nur die Kenntnis des Problems, sondern auch das Wissen darüber, welche Teile zu welchem Zeitpunkt im richtigen Kontext anzuwenden sind, um im Zusammenspiel zur Problemlösung beizutragen.

„Gestützt auf konstruktivistisch geprägte Annahmen zum Lernen sowie zur Gestaltung von multimedialen Lernumgebungen, geht es darum, anhand komplexer und realistischer Problemstellungen systematisch in die Denk- und Arbeitsweise von Experten einzuführen mit dem Ziel, den Erwerb transferfähigen Wissens mit der Herausbildung allgemeiner und fachspezifischer Problemlösestrategien und Lerntechniken zu verknüpfen" (Klauser, 2002, S.3). Es geht hierbei also nicht ausschließlich um den reinen Wissenserwerb, sondern auch um die Fähigkeit, erworbenes Wissen im Bedarfsfall auch auf andere, wenngleich auch ähnliche Praxissituationen anwenden zu können. Dies erfordert vom Lernenden weitere Kompetenzen und Fähigkeiten: *„Damit der Wissenstransfer aus der Lernsituation in die Lösung von Praxisproblemen gelingt, bedarf es insbesondere des Erwerbs von Wissen, von*

Handlungs- und Medienkompetenzen, von Problemlösungsstrategien, die in Expertengemeinschaften Gültigkeit besitzen" (Kriz & Nöbauer, 2002, S.77).

Insbesondere kann problembasiertes Lernen auch als eine Form des situierten Lernens begriffen werden, da ja die Problemstellung stets mit Bezug zu einem realitätsnahen und anwendungsrelevanten Kontext erfolgt.

5.5 Elemente, Werkzeuge und Systeme des CSCL

CSCL hat als Forschungsgebiet zweifelsohne seinen Ursprung im CSCW: *„CSCL basiert auf und nutzt Technologien, Konzepten und Werkzeugen aus dem Bereich der computerunterstützten Zusammenarbeit ... Dabei sind die Grenzen zwischen CSCW fließend. Arbeits- und Lernprozesse gehen häufig ineinander über, Werkzeuge, die für die Projektarbeit verwendet werden, können meist auch in einer instruktionalen Projektarbeit im Rahmen von Bildungsprozessen verwendet werden"* (Seufert & Wessner, 2004, S.127). Eine klare Grenze zwischen CSCL und CSCW und deren Anwendungen existiert also nicht (und ist möglicherweise auch gar nicht wünschenswert).

Dies erklärt die Existenz einer Vielzahl von CSCL-Systemen, die auf der Basis von Groupware entstanden sind (vgl. Stahl, 2002). Dabei wird typischerweise ein CSCW-System als technische Grundlage für die Gruppenarbeit (resp. das Gruppenlernen) verwendet und um pädagogisch-didaktische Methoden ergänzt (vgl. Pfister & Wessner, 2001b, S.252). Und obwohl die technischen Möglichkeiten auch die gleichseitige Verwaltung vieler Benutzer zulassen, findet beim Gruppenlernen (im Allgemeinen genauso wie in CSCL-Systemen im Speziellen) in der Regel eine Beschränkung auf Kleingruppen statt: *„Weil das Lehren und Lernen in großen Gruppen schwierig ist, liegt es nahe, die große Gruppe in mehrere Kleingruppen aufzuteilen, die dann unter wesentliche besseren didaktischen Bedingungen arbeiten können. Das macht jeder Lehrer so, und auch Tutorengruppen an Universitäten verfolgen dasselbe Ziel"* (Effelsberg et al., 2004, S.96).

Die Durchführung kollaborativer Lernprozesse (wie bspw. beim problem-basierten Lernen; s. Kap.5.4.4) erfolgt bevorzugt in Kleingruppen, da diese unter anderem von einem potentiell hohen Maß an Interaktivität profitieren. Mit zunehmender Gruppengröße nimmt der Anteil des Einzelnen an der sozialen Interaktion ab, Kollaborationen können dort nur unter erschwerten Bedingungen aufrecht erhalten und zum Erfolg geführt werden (vgl. Sader, 2002, S.62f; Döring, 2003, S.498). Lernprozesse in Großgruppen werden daher in der Regel in solchen Lehrformen stattfinden, die per se einen eher geringen Grad an Interaktion und Kollaboration aufweisen wie beispielsweise virtuelle Seminare bzw. virtuelle Vorlesungen, die als Videokonferenz an mehrere Standorte übertragen werden (vgl. Effelsberg et al., 2004).

Im Rahmen der vorliegenden Arbeit werden ausnahmslos Kleingruppen betrach-

tet, sodass nachfolgend auch der Schwerpunkt auf solchen CSCL-Systemen liegen wird. Insbesondere liegen diesen Systemen als Bausteine dieselben Elemente der computer-vermittelten Kommunikation zugrunde, die auch im Rahmen von CSCW bereits betrachtet wurden (s. Kap.4.6).

5.5.1 Elemente computer-vermittelter Kommunikation im CSCL

Die grundlegenden Bausteine, aus denen sich die CSCL-Systeme konstruieren lassen, sind – wie auch bei der Groupware – die Elemente der computervermittelten Kommunikation (s. Kap.3.1 und Kap.4.6). In diese Kategorie fallen asynchrone Dienste wie beispielsweise Email und Newsgroups sowie synchrone Dienste wie Instant Messaging, IP-Telefonie (*Voice Over IP*; VoIP) sowie audio- und videobasierte Konferenzsysteme (vgl. Schümmer & Haake, 2004; Appelt, 2004, S.137ff).

In der computerunterstützten Lehre haben sich für diese Elemente verschiedene Einsatzbereiche herausgebildet (vgl. Schümmer & Haake, 2004, S.71ff):

- Themenbezogene Kommunikation:
  Lehrstoffbezogene Diskussionen (bspw. über Newsgroups und Foren) dienen der Erarbeitung von Lerninhalten im Austausch mit anderen Studierenden. Hierbei ist der Einsatz von Moderatoren zu empfehlen, um einerseits mögliche Fehlinformationen zu korrigieren und um andererseits Ergebnisse zusammenzufassen.

- Soziale Kommunikation:
  Für eine funktionierende Interaktion zwischen den Studierenden ist es von Vorteil, dass diese einander kennen (vgl. Döring, 2003, S.521f). Daher sollten Möglichkeiten bereit gestellt werden, über die sich jedes Mitglied einer Lerngruppe oder -gemeinschaft den anderen vorstellen kann. Dies kann beispielsweise mittels einer persönlichen Web-Seite innerhalb einer Lernumgebung erfolgen oder aber auch durch einen kurzen Beitrag in einem Forum bzw. einer Newsgroup.

- Koordinierende Kommunikation:
  Die Bildung von Lerngruppen kann (und sollte) durch ein CSCL-System unterstützt werden, da diese die Grundlage des kollaborativen Lernens darstellen. Im einfachsten Fall kann dies durch entsprechende Beiträge in Foren erfolgen (bspw. in der Form *„Ich suche noch Teammitglieder zur Bearbeitung von Aufgabe x.y - bitte melde Dich bei..."*), es können aber auch spezielle Werkzeuge zur Gruppenbildung zum Einsatz kommen (s. Kap.5.5.2). Auch die Terminplanung für die Durchführung von Phasen synchroner Kooperation kann computerunterstützt durchgeführt werden. In Kleingruppen kann dies im einfachsten Fall per Email erfolgen. Auch der Zugriff auf CSCW-Werkzeuge wie Gruppenkalender kann zur Unterstützung von Lerngruppen herangezogen werden.

Wie bereits bei CSCW existieren auch in CSCL zusätzlich zu den genannten grundlegenden Elementen auch Dienste, die spezielle Aspekte der jeweiligen Kooperationsform (hier: des gemeinsamen Lernens) mittels geeigneter Ansätze und Konzepte unterstützen bzw. fördern. Zuvor soll jedoch die computergestützte Möglichkeit zur Bildung von Lerngruppen als Vorstufe kollaborativen Lernens im Sinne von CSCL betrachtet werden,

5.5.2 Koordinierende Werkzeuge zur Gruppenbildung

Sofern die Gruppenbildung nicht durch Dozenten vorgenommen wird, sondern den Lernenden überlassen bleibt, ist es sinnvoll, wenn die Lernumgebung selber diesen Prozess mittels geeigneter integrierter Werkzeuge initiieren und begleiten kann, denn *„virtuelle Lernumgebungen (Lernplattformen) bieten im Gegensatz zum klassischen Präsenzlernen zunächst allerdings keine Möglichkeiten, durch direkte Kontakte zwischen Lernenden soziale Netzwerke zu bilden"*(Reichling et al., 2004, S.80). Eine Lernplattform mit entsprechender Unterstützung kann die soziale Vernetzung von Lernenden fördern und diese dabei unterstützen, sich zu Lerngruppen zusammen zu schließen.

Solche koordinierenden Werkzeuge können dabei nach unterschiedlichen Strategien vorgehen. Eine einfache Strategie kann sich schlichtweg nach der Verfügbarkeit der Studierenden richten: nach Eingabe eines gewünschten Termins liefert die Lernumgebung eine Übersicht aller zu diesem Zeitpunkt verfügbaren Studierenden und kann diese – automatisch oder nach Bestätigung durch den Suchenden – über das Gesuch informieren, Rückmeldungen verwalten, Alternativen anbieten und letztlich einen Termin für die Durchführung einer Phase synchronen kollaborativen Lernens samt aller Beteiligten festlegen. Dies setzt jedoch voraus, dass alle Teilnehmer ihre relevanten Termine und Freizeiten konsequent und vollständig innerhalb der Lernumgebung pflegen.

Eine andere Vorgehensweise bei der Unterstützung zur Bildung von Lerngruppen kann versuchen, *„Lernende mit ähnlichen oder sich ergänzendem Hintergrund, Interessen oder Bedürfnissen bekannt zu machen. Die zentrale Herausforderung bei der Gestaltung dieser Funktionalität besteht darin, Akteure mit ähnlichen oder komplementären Eigenschaften in der virtuellen Realität einer Lernplattform zusammenzuführen"* (Reichling et al., 2004, S.80). Als Grundlage für die Auswahl passender Teilnehmer müssen diese innerhalb der Lernumgebung modelliert werden, indem Interessen, Qualifikationen, Verhaltensweisen und weitere relevante Eigenschaften erfasst werden. Für jeden Teilnehmer existiert ein Profil innerhalb der Lernumgebung, das für die Gruppenbildung relevante Informationen enthält. Es hat sich gezeigt, dass bei der Auswahl möglichst Übereinstimmungen bzgl. Sprache, Werten und Normen zu berücksichtigen sind (vgl. Reichling et al., 2004, S.84).

Insgesamt stellen diese Werkzeuge sicher einen geeigneten Ansatz dar, um in einem komplett virtuellen Lernszenario Lernende bei der Bildung von Lerngruppen (und ggf. darüber hinaus auch bei der Terminfindung) zu unterstützen. In hypriden Lernsituationen, also solchen mit eher traditionell orientierten Präsenzphasen und

virtuellen Phasen, können sich Lerngruppen auf *natürliche* Weise bilden, also durch in face-to-face-Situationen herbeigeführte soziale Kontakte. In vernetzten System und rein virtuellen Lernszenarien ist dies nicht möglich und muss daher simuliert werden (vgl. Reichling et al., 2004, S.84).

5.5.3 Kooperative Werkzeuge zur Lernprozessunterstützung

Die Unterstützung der eigentlichen Kooperation, also des Lernprozesses per se (einer virtuellen Lernguppe), lässt sich zunächst auf entsprechende Werkzeuge zurückführen, die auch beim kooperativen Arbeiten eingesetzt werden. Private und gemeinsame Arbeitsbereiche können dem Einzelnen bzw. der Gruppe für die Ablage von Dokumenten, URLs, Notizen etc. bereitgestellt werden. Kooperative oder Gruppeneditoren ermöglichen die gemeinsame Bearbeitung von Dokumenten. Je nach Ausprägung handelt es sich dabei um einen gemeinsamen Informationsraum, in welchem jeder Teilnehmer eine beliebige Anzahl von Informationsobjekten zeitgleich zu den anderen Teilnehmern seiner Gruppe bearbeiten kann, oder aber um gemeinsame Informationsobjekte, die von allen Teilnehmern der Gruppe gleichzeitig editiert werden (vgl. Holmer & Jödick, 2004, S.89; Appelt, 2004, S.139; Kap.4.6.2).

Neben dieser eher technisch-basierten Unterstützung der Zusammenarbeit wird im CSCL auch die Unterstützung sozialer Aspekte betrachtet. Ein wichtiger Punkt ist hierbei die *Awareness*, die gegenseitige Wahrnehmung innerhalb der Gruppe als unentbehrliche Grundlage jeglicher Kommunikation und darauf aufbauend Koordination und Kooperation (s. Kap.4.6.1), denn *„wenn die Teilnehmer einer Gruppe nicht an einem Ort zusammen sind und sehen können, was die anderen gerade tun, entsteht ein Defizit in der sozialen Wahrnehmung und daraus folgen Koordinationsprobleme. In diesen Fällen muss die Anwendung dieses Defizit kompensieren, in dem sie explizit folgende Awareness-Informationen anzeigt: Zustand und Kontext einzelner Teilnehmer, Status der Objekte und Prozesse sowie Gruppen- und Einzelaktivitäten"* (Holmer & Jödick, 2004, S.88). Die im Zusammenhang von Awareness notwendige Beobachtung von einzelnen Lernenden wie auch einer ganzen Lerngruppe kann ferner für die Bewertung von Aufgaben und damit verbunden der Zuordnung von Individualleistungen verwendet werden vgl. Appelt, 2004, S.139f.

Zusätzliche Steuermechanismen greifen unterstützend im Falle einer notwendigen Neuausrichtung der Gruppenaktivität ein und ermöglichen Abstimmungen der Mitglieder über den weiteren Verlauf, geben Feedback bezüglich erreichter Arbeitsergebnisse und stellen Testfragen, so dass durch den Tutor eine Überprüfung des Lernstatus im Hinblick auf die zu vermittelnden Lehrinhalte erfolgen kann (vgl. Holmer & Jödick, 2004, S.89f; Appelt, 2004, S.139f).

5.5.4 Virtuelle kooperative Lernräume

Gruppenlernen beinhaltet – neben anderen Elementen wie beispielsweise dem aktiven Diskurs – auch immer die Nutzung gemeinsamer Materialien (wie Skripte, Aufgaben etc.) und Informationen, die innerhalb der Gruppe ausgetauscht und bearbeitet werden. Im Falle von CSCL – speziell beim D-CSCL – müssen diese Ressourcen idealerweise an allen Lernorten und zu jedem Zeitpunkt aktuell bereitstehen. Auch müssen bestimmte vorteilige Aspekte realer Lernsitutationen wie direkte soziale Kontakte und daraus resultierend einfache KoordinationsMöglichkeiten und Aufbau eines Gruppenbewusstseins berücksichtigt werden. Diese Forderungen lassen sich mittels *virtueller Lernräume* erfüllen: *„Virtuelle Lernräume dagegen versuchen, mit Hilfe einer Software-Umgebung die Eigenschaften und Vorteile einer realen Lernumgebung so weit wie möglich nachzubilden, um ein verteiltes, ortsunabhängiges Lernen zu unterstützen. Virtuelle kooperative Lernräume ... bieten darüber hinaus eine funktionale Unterstützung in den Bereichen Kommunikation, Koordination und Kooperation"* (Dawabi, 2004, S.118f).

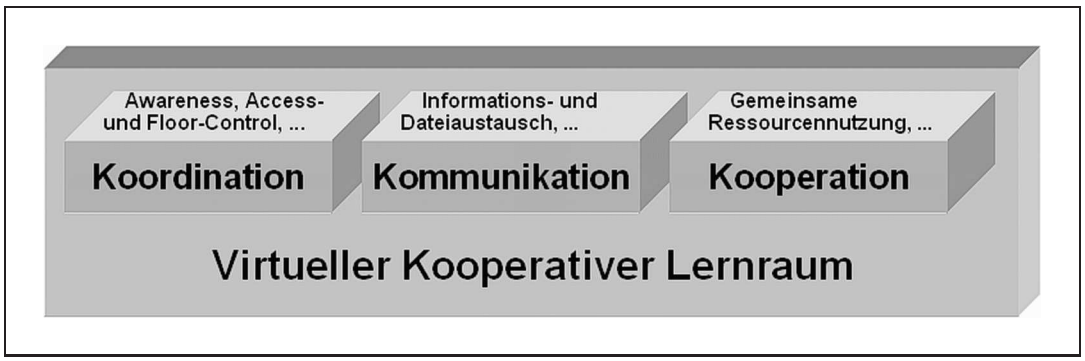

Abb. 5.5: Funktionale Bereiche eines virtuellen kooperativen Lernraums (Dawabi, 2004, S.119)

Ein virtueller kooperativer Lernraum umfasst also – wie in Abbildung 5.5 skizziert – die zielgerichtete Bündelung von vorhergehend genannten CSCL-Elementen und -Werkzeugen und stellt einen ganzheitlichen Ansatz zur computer-gestützten Realisierung und Förderung kollaborativen Lernens dar.

5.5.5 CSCL-Plattformen

CSCL-Plattformen sind Lernumgebungen, die sich durch einen hohen Integrationsgrad der dort verfügbaren CSCL-Werkzeuge auszeichnen. In der Regel handelt es sich um Web-Portale, bei denen die einzelnen Komponenten eng gekoppelt sind und ineinander greifen, so dass diese einen gewissermaßen ganzheitlichen Ansatz zur Unterstützung von netzwerkbasierten, kooperativen Lernprozessen bilden (vgl. Seibt, 2001, S.9f; Hagenhoff et al., 2001; Appelt, 2004, S.137f).

Aus diesem hohen Integrationsgrad resultiert ein üblicherweise großer Leistungsumfang: *„Abgesehen von den Möglichkeiten zur Unterstützung der originären Aufgaben können Lernplattformen eine Reihe von zusätzlichen Features aufweisen"*

(Hagenhoff et al., 2001, S.2). Neben den üblichen Werkzeugen zur Unterstützung von Kommunikation, Koordination und Kooperation (s. Kap.5.5.1, 5.5.2 und 5.5.3) besitzen CSCL-Plattformen beispielsweise auch Evaluationswerkzeuge zur Ermittlung von Aktivitäten und Lernerfolgen der Studierenden:

- Selbsttests, die im Anschluss an absolvierte Lerneinheiten von den Teilnehmern durchgeführt werden, geben Auskunft über den Lernerfolg.
- Studenten- und Kursmonitoring helfen dem Tutor, zu ermitteln, welche Studenten welche Lerneinheiten wann mit welchem Erfolg bearbeitet haben.
- In einige Plattformen sind spezielle Werkzeuge integriert, die Tutoren bei der der Bewertung von Arbeitsergebnissen unterstützen.
- Um bei Gruppenleistungen die Beiträge der einzelnen Gruppenmitglieder ermitteln zu können, bieten einige Plattformen Werkzeuge an, mit denen das Interaktionsverhalten einer Gruppe beobachtet und ausgewertet werden kann.

Durch die Integration von Autorenwerkzeugen in eine Lernplattform werden Autoren bei der Erstellung und Bereitstellung von Lernmaterialien unterstützt, indem geeignete Kursvorlagen innerhalb der Lernplattform verwaltet und bei Bedarf abgerufen werden können. Darüber hinaus können auch durchzuführende Evaluationsschritte berücksichtigt werden, die von den Autoren entsprechend auszuarbeiten sind.

Administrative Werkzeuge übernehmen die Verwaltung von Benutzern, Benutzergruppen und Ressourcen, unterstützen bei der Vergabe und Verwaltung von Zugriffsrechten und erledigen die Registrierung neuer Benutzer sowie die Authentifikation bereits vorhandener Benutzer, wenn diese sich beispielsweise an einer Lernplattform anmelden.

Insgesamt stellt sich der Übergang zwischen den in Kapitel 5.5.4 behandelten virtuellen kooperativen Lernräumen und den hier betrachteten CSCL-Plattformen als fließend dar, so dass eine trennscharfe Unterscheidung kaum möglich ist, jedoch weisen die Lernplattformen neben einem vergleichsweise hohen Integrationsgrad der angebotenen Werkzeuge auch einen entsprechenden Leistungsumfang auf. Hinzu kommt eine umfassende Behandlung von Lernprozessen innerhalb von Lernplattformen: die Kombination von unterschiedlichen Werkzeugtypen bietet einerseits Lerngruppen Unterstützung bei der Durchführung ihrer jeweiligen Lernprozesse und ermöglicht andererseits die Vorbereitung der Lernszenarien und die Administration von damit assoziierten Ressourcen jeglicher Art (Lernende, Lerngruppen, Lernmaterialien etc.). Dadurch wird eine umfassende Betrachtung von Lernprozessen aus verschiedenen, sich ergänzenden Blickwinkeln (bspw. Lernende, Dozenten, Administratoren) ermöglicht.

Auf eine nährere Betrachtung spezifischer CSCL-Systeme an dieser Stelle wird bewusst verzichtet. Dieses würde aufgrund der Vielzahl inzwischen existierender

Systeme entweder den Rahmen der vorliegenden Arbeit sprengen oder eine eng begrenzte Auswahl mit nur gereingem Aussagewert darstellen. Stattdessen sei der interessierte Leser auf weiterführende Literatur verwiesen.

5.6 Dimensionen des CSCL

Die in der Abbildung 5.3 ersichtliche Einordnung von CSCL ist zugegebenermaßen nur recht grob. Um bei der inzwischen gegebenen Vielzahl von CSCL-Systemen (vgl. McManus, 1997, S.8; Wessner, 2001, S.207-219; Haake et al., 2004, Kap.2) einzelne CSCL-Applikationen genauer positionieren zu können, bedarf es daher der Auswahl und Festlegung geeigneter Dimensionen, anhand derer sich CSCL-Systeme klassifizieren lassen (s. Kap.4.7). *„Jede Positionierung entlang dieser Dimensionen weist bestimmte technische und pädagogisch-psychologische Besonderheiten auf und hat Einfluß auf das Design einer CSCL-Umgebung"* (Wessner, 2001, S.203).

5.6.1 Ort und Zeit

Analog zu den Klassifikationsansätzen von Groupware (s. Kap.4.7) kann eine grundlegende Einteilung von CSCL-Systemen anhand der räumlichen und zeitlichen Verteilung der Teilnehmer durchgeführt werden:

- Die zeitliche Dimension unterscheidet synchrones CSCL von asynchronem CSCL.
- Die räumliche Dimension entscheidet darüber, ob sich die Teilnehmer am selben Ort oder an verschiedenen Orten befinden.

In Anlehung an Abbildung 4.4 ergibt sich eine Raum-Zeit-Matrix (s. Tab.5.1).

Tab. 5.1: Raum-Zeit-Matrix für CSCL

	Gleicher Ort	Verschiedener Ort
Gleiche Zeit (synchron)	z.B. Computer-unterstütztes Klassenzimmer	z.B. Televorlesung
Verschiedene Zeit (asynchron)	z.B. Schwarzes Brett	z.B. Diskussionsforum

(vgl. Haake et al., 2004, S.2)

Da CSCL in der Regel unter Einsatz von miteinander vernetzten Rechnersystemen stattfindet, ist räumlich verteiltes CSCL der häufigere Anwendungsfall. Dies wird in der Literatur auch als D-CSCL (*Distributed CSCL*; dt.: verteiltes CSCL) bezeichnet: *„Da in vielen Anwendungskontexten alle oder einige der Beteiligten*

örtlich verteilt sind, spricht man auch von D-CSCL (das D steht hier für Distributed)" (Pfister, 2000, S.228). Das im Rahmen der vorliegenden Arbeit zum Einsatz kommende System ist genaugenommen als D-CSCL-System zu bezeichnen. Im Laufe der weiteren Ausführungen wird weiterhin von CSCL die Rede sein, eine Differenzierung zwischen CSCL und D-CSCL wird jedoch nur dort stattfinden, wo sie unabdingbar ist.

5.6.2 Symmetrie

Unterschiede zwischen den Wissensniveaus der Beteiligten werden mit der Dimension *Symmetrie* zum Ausdruck gebracht: *„Tauschen Personen mit vergleichbaren aber heterogenen Wissensniveaus ihr Wissen aus oder liegt ein starkes Wissensgefälle vor?"* (Haake et al., 2004, S.3). Anhand dieser Fragestellung erfolgt eine Differenzierung zwischem symmetrischem CSCL, bei welchem alle Beteiligten vergleichbare Wissensniveaus vorweisen, und asymmetrischem CSCL mit entsprechend unausgeglichenen Wissensniveaus.

5.6.3 Direktivität

Diese Dimension repräsentiert den Grad der Anleitung einer Lerngruppe während des Lernprozesses dar. Anhand dieser Dimension lassen sich selbstorganisierende Lerneinheiten von solchen unterscheiden, die durch Personen oder Programme angeleitet werden.

5.6.4 Dauer

Diese Dimension differenziert zwischen spontanen, kurzlebigen Lerngruppen einerseits und langfristig orientierten Lerngruppen andererseits. Letztere existieren über einen längeren Zeitraum und treffen sich währenddessen wiederholt zur Bearbeitung und Vertiefung von für sie relevanten Lernstoffen: *„Darunter fallen Seminargruppen oder Lerngemeinschaften, deren Mitglieder sich über Monate oder Jahre gemeinsam fortbilden. Aus technischer Sicht spielen hier die Sitzungsverwaltung sowie die Bereitstellung persistenter Kooperationsräume eine bedeutende Rolle"* (Wessner & Pfister, 2001, S.204). Für die vorliegende Arbeit haben langfristig orientierte Lerngruppen keine (oder nur geringe) Relevanz, es werden vorrangig Lerngruppen mit einer Lebensdauer von wenigen Stunden betrachtet (vg. Kap.7.1.2.3).

5.6.5 Wissensziel

Mittels dieser Dimension werden die Fragen *„Soll am Ende des Lernprozesses jeder Beteiligte einzeln oder die Gruppe als Ganzes über das Wissen verfügen?"* und

„Geht es um das Zusammentragen von Informationen, das Anwenden und Vertiefen von Kompetenzen oder das Herausbilden eines gemeinsamen Verständnisses?" (Haake et al., 2004, S.3) beantwortet werden. Es erfolgt also anhand dieser Ziele eine Unterscheidung zwischem individuellem Wissenserwerb und Wissenserwerb in der Gruppe als Gesamtheit.

5.6.6 Gruppengröße

Einteilungen anhand der Größe von Lerngruppen reichen von Lernpaaren bis hin zu Lerngemeinschaften mit prinzipiell beliebig vielen Teilnehmern. Aus der Gruppengröße resultieren mögliche Lernmethoden und Interaktionsformen innerhalb der Lerngruppe.

Betrachtet man CSCL anhand dieser Dimensionen, so wird ersichtlich, dass diese Lernform viele unterschiedliche Varianten vom einfachen Email-Kontakt zwischen Lernendem und Dozenten bis hin zu komplexen, langfristig ausgerichteten Lernszenarien mit vielen Beteiligten beinhaltet.

5.7 Unterstützung kollaborativen Lernens

Lerngruppen benötigen während ihres Lernprozesses in CSCL-Szenarien jedoch mehr als nur eine funktionierende Anwendung, die die technischen Aspekte des jeweiligen Lernszenarios unterstützt. Gerade die Förderung des kollaborativen Lernprozesses stellt eine der zentralen Fragestellungen von CSCL dar (vgl. Haake et al., 2004, S.2).

5.7.1 Probleme virtueller Lerngruppen

Trotz der genannten Vorteile kollaborativen Lernens (s. Kap.2.2.6) sehen sich auch virtuelle Lerngruppen und ihre Mitglieder neben inhaltlichen Problemen, die aus der zu bearbeitenden Aufgabe resultieren, vielen der üblichen Schwierigkeiten gemeinschaftlichen Lernens ausgesetzt: *„Denn auch über das Netz beobachten wir alle – funktionalen wie dysfunktionalen – Gruppenprozesse (wie z.B. die mehr oder weniger ausgeprägte Identifikation mit der Gruppe und dem Gruppenergebnis, die Herausbildung von Gruppennormen und -strukturen, die Verstärkung der Lernmotivation des Einzelnen durch die Gruppe, aber auch: die »schweigende Mehrheit«, die Profilierung Einzelner, vorschnelles Aufteilen von anstehenden Aufgaben statt gemeinsames Erarbeiten und Diskutieren u.v.a.m.)"* (Kerres, 2001, S.297).

Insbesondere werden oftmals unstrukturiertes Vorgehen und fehlende Koordination bei der Aufgabenbearbeitung beobachtet, die mit mangelnder Erfahrung in Sachen Gruppenarbeit begründet werden können. Hinzu kommen einzelne Mitglieder, die bei der Gruppenarbeit eher Zurückhaltung an den Tag legen (sog. Trittbrettfahrer), und ein mangelhaftes Zeitmanagement, was dazu führt, dass wertvolle Zeit anstatt

für die eigentliche Aufgabenbearbeitung nunmehr für unwichtige Tätigkeiten und Diskussionen aufgebracht wird (vgl. Breuer, 2001, S.13; Schenk, 2004, S.208f).

Darüber hinaus kann auch die in CSCL-Systemen zum Einsatz kommende computer-vermittelte Kommunikation als Ursache für eine Vielzahl von Problemen angesehen werden: einerseits führt die mit computer-vermittelter Kommunikation üblicherweise einhergehende Kanalreduktion (vgl. Döring, 2003, S.149ff) zu einer Verarmung der Kommunikation, da wichtige Elemente (wie bspw. Gestik und Mimik) herausgefiltert werden, sowie zu einer verringerten Wahrnehmung der Kommunikationspartner, andererseits sehen sich die Kommunikationsteilnehmer oftmals einer Informationsflut ausgesetzt, verursacht durch ein Überangebot von Informationsmöglichkeiten (in Form von Emails, Chat, Instant Messaging etc.). In Folge dessen wird auch die Zuordnung von zusammengehörenden Nachrichten erschwert (vgl. Breuer, 2001, S.12ff).

5.7.2 Betreuung durch Tutoren

Zur Lösung der genannten Probleme kommen für gewöhnlich Tutoren zum Einsatz: *„Der Begriff »Tutor« stammt ursprünglich aus der angelsächsischen Hochschulpraxis. TutorInnen sind (höhersemestrige) Studierende, die andere Studierende in ihrem Studium beraten und bei der Erarbeitung von Themengebieten unterstützen"* (Schlienger-Merki & Schauer, 2004, S.220). Im Kontext von E-Learning und CSCL wird der Tutor in der Literatur auch als *E-Tutor*, *Online-Tutor* oder *Teletutor* bezeichnet. Der Vorgang der Unterstützung durch einen Tutor wird *Tutoring* (bzw. *E-Tutoring*, *Online-Tutoring* und *Teletutoring*) genannt. Mit dem Tutoring artverwandt sind Konzepte wie das Mentoring, das Coaching oder auch die Lernberatung (vgl. Wilbers, 2001, S.29; Schlienger-Merki & Schauer, 2004, S.219f), die oftmals auch synonyme Verwendung finden.

Ergänzend zu der Definition von Schlienger-Merki & Schauer (2004) kann auch ein Lehrender die Rolle eines Tutors übernehmen, denn wichtiger als der Personenkreis sind wünschenswerte Eigenschaften und wahrzunehmende Aufgaben: *„Tutoren haben die Aufgabe, die Lernenden bei der Bearbeitung der Lernaufgabe zu motivieren, individuelle Überlegungen und Lösungsvorschläge nachzuvollziehen und Hilfestellungen zu geben. Tele-Tutoren unterstützen dieWissensaneignung der Lernenden, in ihrer Moderatorenfunktion sind sie verantwortlich für ein vertrautes Kursklima, einen offenen Kommunikationsstil und die Formulierung von klaren Vorgaben, die zugleich auch langsam an die Lernenden delegiert werden sollen. Als Organisatoren sind sie für die Einhaltung der Termine verantwortlich und bieten zugleich den Lernenden technischen Support"* (Petschenka et al., 2004, S.15).

Folglich ist ein Tutor gleichsam Moderator wie inhaltlicher Experte, denn *„der Tutor muss über die notwendige fachliche, methodische und technische Kompetenz verfügen und bei der Betreuung der Teilnehmer die notwendige methodische Vielfalt nutzen"* (Weidmann, 2001, S.2). Moderation bedeutet in diesem Zusammenhang die Berücksichtigung verschiedenartiger Funktionen (vgl. Schenk, 2004, S.209):

- Die aufgabenbezogene Funktion beinhaltet die Planung von Zielen, Leistungsumfang und Arbeitsschritten, um die Bearbeitung der zugrundeliegenden Sachaufgabe durch die Gruppe zu realisieren.
- Die gruppenbezogene Funktion schafft durch das Festlegen von Gruppenwerten, -normen und -regeln eine Basis für eine funktionierende Zusammenarbeit innerhalb der Gruppe.
- Die personenbezogene Funktion beinhaltet die Unterstützung jedes einzelnen Gruppenmitglieds beispielsweise in Form von Coaching (vgl. Schlienger-Merki & Schauer, 2004).

Im Kontext von CSCL beinhaltet die Moderation darüber hinaus auch die Wahrnehmung von technischen Hilfestellungen bei Problemen in der Bedienung des jeweiligen CSCL-Sytems durch die Teilnehmer (vgl. Schenk, 2004, S.218). Die Ziele einer effektiven Moderation liegen also in der Integration aller Mitglieder in die (Lern-)Gruppe sowie deren Überführung in die Selbständigkeit.

Solche Aktivitäten, die sich auf den eigentlichen Inhalt der Sachaufgabe beziehen, sind hiervon ausgenommen; sie gehören nicht zu den Aufgaben des Moderators, sondern sind dem Lehrenden zuzuordnen. Zusammenfassend verlangt die tutorielle Betreuung von Lerngruppen im CSCL außer Moderationstätigkeiten auch fachlich-inhaltliches Expertenwissen sowie die Anwendung geeigneter pädagogisch-didaktischer Maßnahmen, um in konkreten Problemsituation angemessene Hilfestellung leisten zu können. Dazu gehören auch Rückmeldungen durch den Tutor an die Lernenden nach absolvierten Arbeitsschritten (vgl. Kerres, 2001, S.297f; Petschenka et al., 2004, S.15f).

Ergänzend zu den bereits in Kapitel 5.5.5 genannten Komponenten empfiehlt es sich, geeignete Funktionen zur Unterstützung von Tutoring-Maßnahmen in Lernplattformen zu integrieren: *„Tutoring-Systeme sind modular aufgebaute Systeme, die speziell die Aktivitäten von Tutoren bzw. Betreuern methodisch unterstützen. Der Bedarf an derartigen Systemen resultiert aus der bei E-Learning-Maßnahmen in der Praxis gemachten, außerordentlich wichtigen Erfahrung, dass erfolgreiche EL-Maßnahmen immer die Beteiligung von menschlichen Betreuern bzw. Tutoren voraussetzen. Diese Systeme müssen an die individuellen Randbedingungen einer E-Learning-Maßnahme angepasst werden"* (Seibt, 2001, S.9).

In einigen Lernplattformen wird bereits der Versuch unternommen, wesentliche Funktionen des Tutoring software-technisch in das jeweilige CSCL-System zu integrieren und auf diese Weise menschliche Tutoren sukzessive durch künstliche, sogenannte *virtuelle Tutoren* zu ersetzen: *„Mit der Entwicklung multimedialer Lernumgebungen werden das Tutoring bzw. tutorielle Funktionen in zunehmendem Maße auch von computergestützten »intelligenten tutoriellen Systemen« übernommen"* (Klauser, 2002, S.12). Hier knüpft das CSCL-System des Projekts *VitaminL* an.

Kapitel 6

Intelligente tutorielle Systeme

Als *Intelligente Tutorielle Systeme* (ITS) bezeichnet man gemeinhin spezielle E-Learning-Angebote, die anhand der erfolgten Bedienschritte eines Benutzers dessen aktuelle Lernsituation hinsichtlich vorhandener Kompetenzen (bzw. Kompetenzdefizite) analysieren und daraufhin Entscheidungen bezüglich des tutoriellen Angebots treffen (vgl. Kerres, 2001, S.71). Spezielle ITS, deren Fokus auf der Unterstützung kollaborativer Lerngruppen liegt, werden auch *Intelligent Collaborative Learning System* (ICLS) genannt, als Fusion aus CSCL und ITS (vgl. Soller, 2001, S.41). Der Schwerpunkt der vorliegenden Arbeit ist auf ebensolche ICLS ausgerichtet, die in dieser Arbeit als Spezialfall von ITS anzusehen und eben diesen zuzuordnen sind.

6.1 Ursprung von ITS

Ihren Ursprung haben ITS in den klassischen CBTs (s. Kap.5.3.4.1) beziehungsweise in deren Defiziten: Wo ein Lehrer bemüht ist, sein Unterrichtsverhalten an aktuelle Lernprozesse anzupassen, versagen die klassischen Lernprogramme, indem sie *„den Lerner zwangen, den programmierten Wegen der Maschine zu folgen"* (Kerres, 2001, S.70). Ein Eintreten in den pädagogischen Dialog, das das für den fruchtbaren Lernprozess unabdingbare Wechselspiel von Interaktionen zwischen Lehrer und Lerner darstellt, unterbleibt weitestgehend, da die dazu notwendigen diagnostischen Fähigkeiten der CBTs nur gering bis gar nicht ausgeprägt sind, denn *„bei klassischen CBT-Anwendungen beschränkt sich die Diagnose folglich auf die Auswertung von Testantworten, die im Anschluss an Informationseinheiten präsentiert werden. Mit diesem Vorgehen liegt natürlich keine Diagnose im eigentlichen Sinne vor: Es wird lediglich festgestellt, ob ein Fehler vorliegt oder nicht"* (Kerres, 2001, S.70).

Seit den 1980er Jahren kommen Techniken aus dem Bereich der Künstlichen Intelligenz (KI) zum Einsatz, um besagtes Defizit zu kompensieren und adaptive Systeme zu erstellen, mit deren Hilfe individualisiertes Lernen ermöglicht wird

(vgl. SCHULMEISTER, 1997, S.199; KERRES, 2001, S.71). Die resultierenden Systeme, von SCHULMEISTER (1997) nicht ohne jede Kritik als *„denkende, sprechende und Sprache verstehende Programme"* (SCHULMEISTER, 1997, S.177) bezeichnet, bestehen in der Regel aus folgenden vier Komponenten (vgl. SCHULMEISTER, 1997, S.182):

- Modellierung eines Wissensgebietes,
- Modell des Lernenden,
- Modell pädagogischer Strategien und
- Komponente für Kommunikation.

Vor einer näheren Charakterisierung dieser Komponenten erfolgt zunächst eine Übersicht über diejenigen Software-Techniken, die in ITS und ihren Komponenten typischerweise Verwendung finden.

6.2 Konzepte der Künstlichen Intelligenz

Die Künstliche Intelligenz (KI) ist ein interdisziplinäres Forschungsgebiet, an dem neben der Informatik auch die Psychologie, die Biologie, die Linguistik sowie die Mathematik beteiligt sind. Man befasst sich in der KI mit Denkprozessen, wie sie vornehmlich im menschlichen Gehirn stattfinden, und versucht, diese zum besseren Verständnis mittels Computersystemen nachzubilden und auf diese Weise ein Gehirn (in Teilen oder als Ganzes) zu simulieren. Darüber hinaus werden Ansätze erforscht, mit denen Computerprogramme *intelligenter* gemacht werden, indem menschliche Problemlösungsfähigkeiten zur Anwendung kommen.

6.2.1 Wissensrepräsentation

Die Darstellung von Wissen bildet die Grundlage vieler Verfahren der KI. Mit geeigneten Mitteln werden sowohl explizites als auch deklaratives Wissen erfasst, modelliert und als Teil eines Computersystems gespeichert. Explizites oder prozedurales Wissen (vgl. SCHULMEISTER, 1997, S.182) umfasst die reinen Daten eines Problems oder eines Szenarios, deklaratives Wissen ist Wissen, das mittels geeigneter Beschreibungen aus bereits bekanntem Wissen hergeleitet werden kann. Ergänzt werden diese beiden Formen von Wissen üblicherweise um Meta-Informationen, anhand derer unter Einsatz von Wissen und dessen Verknüpfungen Problemlösungen generiert werden können (vgl. DORN & GOTTLOB, 1999, S.977).

Ein wichtiges Werkzeug zur Modellierung von Wissen bietet die Prädikatenlogik erster Ordnung, die eine auf Computersystemen ausführbare Umsetzung in der Programmiersprache *Prolog* besitzt (vgl. RUSSELL & NORVIG, 2004, Kap.8). Eine

andere Art der Darstellung von Wissen findet sich in der *Objektorientierten Wissensrepräsentation*, bei der eine Wissensbasis durch Objekte gestaltet wird (vgl. Dorn & Gottlob, 1999, S.985; Russell & Norvig, 2004, Kap.10.2). Wann immer Unsicherheiten berücksichtigt oder Näherungen verwendet werden müssen, kommen Ansätze wie die Bayes'sche Theorie (vgl. Dorn & Gottlob, 1999, S.982; Russell & Norvig, 2004, Kap.13) oder auch die *Fuzzy Logik* (vgl. Aliev et al., 2000, Kap.3; Russell & Norvig, 2004, S.646f) zum Einsatz.

6.2.2 Wissensbasierte Systeme

Wissensbasierte Systeme setzen sich typischerweise aus einer Wissensbasis und einer Wissensverarbeitungskomponente zusammen. Die Wissensbasis beschreibt das Wissen eines Anwendungsbereiches mittels einer Menge von Wissenseinheiten. Dabei wird üblicherweise differenziert zwischen Wissen, das für ein Fach allgemeingültig ist (taxonomisches Wissen), sowie Fakten und Annahmen für ein konkretes Problem (assertionales Wissen).

In der Wissensverarbeitungskomponente kann aus vorhandenem Wissen neues Wissen generiert werden. Ein in diese Komponente eingebettetes Inferenzsystem stellt allgemein gültige Mechanismen für das Schließen (d.h. für das Ableiten neuen Wissens aus bereits bekanntem Wissen) zur Verfügung, ein Meta-Prozessor ergänzt die Inferenzkompenente um neue Regeln, die aus vorhandenen Regeln ableitbar sind (Meta-Schließen; vgl. Russell & Norvig, 2004, S.238). Die software-technische Realisierung von Wissensverarbeitungskomponenten erfolgt oftmals mittels sogenannter KI-Sprachen wie Prolog oder LISP. Schnittstellen zum Benutzer, zu Datenbanken und zu technischen Prozessen runden ein wissensbasiertes System ab (s. Abb.6.1).

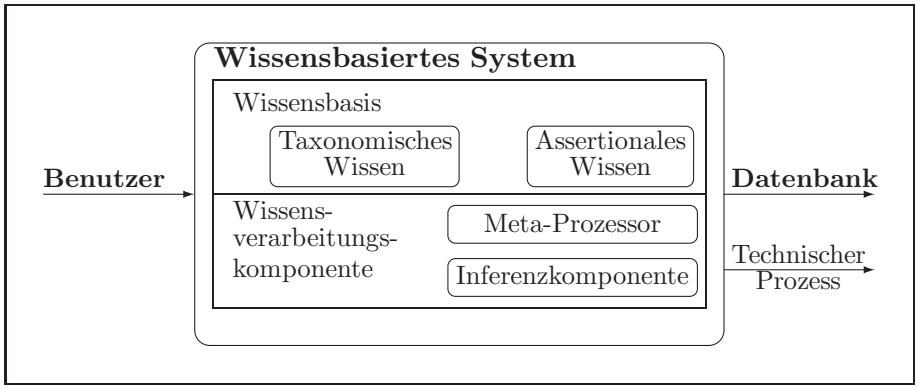

Abb. 6.1: Architektur eines wissensbasierten Systems
(Dorn & Gottlob, 1999, S.978)

6.2.3 Expertensysteme

Es handelt sich bei Expertensystemen um Wissensbasierte Systeme, die in der Lage sind, *„Menschen in relativ komplexen Aufgabenbereichen zu ersetzen und Wis-*

sen, Erfahrung und in besonderem Maß logisches Denken sowie die Fähigkeit des Schulssfolgerns erfordern" (Ferber, 2001, S.22). Ein Wissensbasiertes System samt zugehöriger Benutzerschnittstelle wird erweitert um eine Komponente, über die ein menschlicher Experte dem System weiteres Wissen hinzufügen kann (Wissenserwerbkomponente), und um eine Erklärungskomponente, anhand derer das Expertensystem seine Lösungen dem Anwender darlegt. Die Gesamtarchitektur eines Expertensystems ist in Abbildung 6.2 dargestellt.

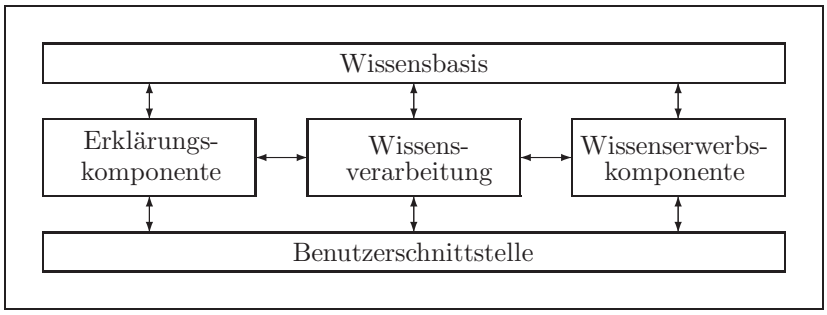

Abb. 6.2: Architektur eines Expertensystems
(Dorn & Gottlob, 1999, S.979)

6.2.4 Wissensbasierte Suche

Der Schwerpunkt dieses Teilgebiets der KI liegt in der Erforschung effizienter Suchverfahren zur Beschleunigung von komplexen Problemlösungsprozessen. Wann immer Verfahren wie Breiten- oder Tiefensuche (Backtracking) nicht geeignet sind, kommen Algorithmen zum Einsatz, die auf Näherungen (Approximationen), Heuristiken (erfahrungsbasierte Modelle), Statistiken oder Zufall beruhen.

Populäre Vertreter sind beispielsweise das *Simulated Annealing* (Simuliertes Abkühlen), das auf physikalischen Effekten beim Erwärmen und Abkühlen von Stoffen beruht (vgl. Dorn & Gottlob, 1999, S.990; Aliev et al., 2000, S.164f; Russell & Norvig, 2004, S.155), oder Genetischen Algorithmen, bei denen natürliche, auf Chromosomen basierende Prozesse wie Mutation und Kreuzung in iterativen Schritten angewandt neue Chromosomen hervorbringen. Diese werden anschließend mittels einer Fitnessfunktion bewertet, bevor eine weitere Iteration durchgeführt wird (vgl. Dorn & Gottlob, 1999, S.991; Aliev et al., 2000, Kap.5).

6.2.5 Maschinelles Lernen

Das Maschinelle Lernen erforscht Konzepte, mit denen neues Wissen aufgenommen und bereits vorhandenes Wissen selbständig reorganisiert werden kann. Das künstliche System lernt anhand von Erfahrungen: Beispiele werden zur Erweiterung der Wissensbasis eingesetzt, in dem das zu trainierende System Gesetzmäßigkeiten in den Lerndaten erkennt. Man differenziert in der Lerntheorie verschiedene Ansätze

(unüberwachtes Lernen, überwachtes Lernen, verstärkendes Lernen und Lernen durch Analogien), die in der KI mittels unterschiedlicher Techniken realisiert werden.

Die Technik des induktiven Schließens leitet allgemeines Wissen aus speziellen beobachteten Fällen ab. Anhand möglichst vieler Positiv- und Negativbeispiele wird ein System angelernt. Es muss charakteristische Eigenschaften finden, um auf diese Weise abstrakteres Wissen selbst zu formulieren. Typische Anwendungen sind Entscheidungsbäume auf Basis von Produktionsregeln (bspw. Regeln zur Kreditvergabe in Banken).

Fallbasiertes Schließen (*Case-based reasoning*; CBR) fasst die charakteristischen Daten eines Problems und dessen Lösung zu einem Fall zusammen und speichert diesen – zusammen mit anderen Fällen – in einer Fallsbasis. Die Lösung eines neuen Problems erfolgt durch Suchen ähnlicher Fälle, wobei der Vergleich anhand eines geeigneten, zu definierenden Ähnlichkeitsmaßes durchgeführt wird. In Kölle (2007) wird diese Technik verwendet, um Hilfe in bestimmten Problemsituationen bei der objektorientierten Programmierung in Java mittels geeigneter Beispiele zu offerieren.

Ein Neuronales Netz kann als Simulation des menschlichen Gehirns verstanden werden: In Schichten angeordnete Neuronen erhalten Reize über Eingabekanäle, woraus sich für jedes Neuron ein Erregungszustand ergibt, welcher über einen Ausgabekanal abgegeben wird. Dieser Ausgabekanal kann mit den Eingabekanälen weiterer Neuronen einer nachfolgenden Schicht verbunden sein. Das Training eines Neuronalen Netzes erfolgt anhand von Beispielen, so dass bei einer gegebenen Eingabemenge eine gewünschte Ausgabemenge berechnet wird. Neuronale Netze werden bevorzugt zur Erkennung von Mustern eingesetzt.

6.2.6 Wahrscheinlichkeitsbasierte Ansätze

Falls das Wissen eines KI-Systems nur unvollständig vorliegt oder mit Unsicherheiten behaftet ist, kommen heutzutage Ansätze zur Anwendung, die auf der Wahrscheinlichkeitstheorie basieren. Wissen wird dabei mit Wahrscheinlichkeiten versehen (in der Literatur taucht in diesem Zusammenhang auch der Begriff des Glaubensgrades auf; vgl. Russell & Norvig, 2004, S.571), so dass die Modellierung unsicheren Wissens ermöglicht wird. Unsicherheit kann aus folgenden Ursachen resultieren (vgl. Russell & Norvig, 2004, S.571; Dorn & Gottlob, 1999, S.981f):

- Der Aufwand für die vollständige und exakte Erfassung sämtlichen Wissens ist unter ökonomische Aspekten schlichtweg nicht gerechtfertigt.

- Eine vollständige Theorie eines bestimmten Wissensgebietes existiert überhaupt nicht, so dass eine vollständige Modellierung bei derzeitigem Kenntnisstand im Grunde genommen unmöglich ist.

- Das Regelwerk eines Wissensgebietes ist zwar vollständig, aber die Daten eines konkreten Falls konnten (noch) nicht vollständig erfasst werden – oder können überhaupt nicht vollständig erfasst werden.

Es existieren mittlerweile diverse Konzepte, die sich beim Schließen unter Unsicherheit gemäß den Gesetzen der Wahrscheinlichkeitstheorie bewährt haben. Nachfolgend werden einige der gängigen Konzepte des Schließens unter Unsicherheit, die in einer Vielzahl von Intelligenten Tutoriellen Systemen zum Einsatz kommen, skizziert, um somit zum besseren Verständnis des Arbeitsprinzips dieser Systeme beizutragen.

6.2.6.1 Bayes'sche Netzwerke

Ein Bayes'sches Netzwerk (BN; *Bayesian Network* oder auch *Belief Network*) ist ein gerichteter azyklischer Graph, dessen Knoten Zufallsvariablen sind. Die Kanten zwischen je zwei Knoten beschreiben die bedingten Abhängigkeiten zwischen den Knoten beziehungsweise zwischen den zugehörigen Variablen. Auf diese Weise wird jedem Knoten X_i des Bayes'schen Netzwerks eine bedingte Wahrscheinlichkeitsverteilung $P(X_i|Eltern(X_i))$ zugeordnet, die zum Ausdruck bringt, inwiefern die zum Knoten gehörige Variable X_i von den Variablen der jeweiligen Elternknoten abhängt. Die Beschreibung der bedingten Wahrscheinlichkeit erfolgt innerhalb des Netzes typischerweise in Form von Bedingten Wahrheitstabellen (BWT), die in unmittelbarer Nähe des jeweiligen Knotens notiert werden (s. Abb.6.3).

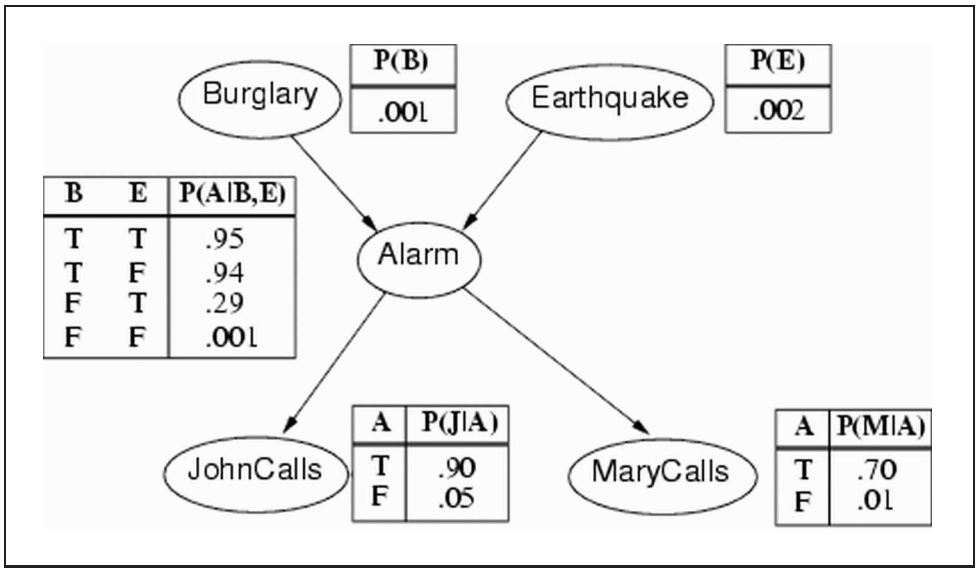

Abb. 6.3: Beispiel eines Bayes'schen Netzwerks
(Russell & Norvig, 2004, S.607)

Mittels eines Bayes'schen Netzes lässt sich auf diese Weise eine Domäne vollständig beschreiben, so dass die Wahrscheinlichkeit für jedes Ereignis anhand der im Netz-

werk hinterlegten Informationen errechnet werden kann:

$$\begin{aligned} P(X_1 = x_1 \wedge \ldots \wedge X_n = x_n) &= P(x_1, \ldots, x_n) \\ &= \prod_{i=1}^{n} P(x_i | eltern(X_i)) \end{aligned}$$

In dieser Formel repräsentiert $eltern(X_i)$ die spezifischen Werte der Variablen in $Eltern(X_i)$.

Darüber hinaus ermöglicht ein Bayes'sches Netz auch das Schließen unter Unsicherheit: *„Die grundlegende Aufgabe jedes probabilistischen Inferenzsystems ist, die bedingte Wahrscheinlichkeitsverteilung für eine Menge von Abfragevariablen für ein bestimmtes Ereignis zu berechnen – d.h. eine Zuweisung von Werten an eine Menge von Evidenzvariablen"* (Russell & Norvig, 2004, S.619). Die theoretische Grundlage dafür bildet das sogenannte *Bayes-Theorem* (oder auch Satz von Bayes; benannt nach Thomas Bayes[1]), das angibt, wie man mit bedingten Wahrscheinlichkeiten rechnet:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Damit erlaubt der Satz von Bayes das Umkehren von Schlussfolgerungen: Anstatt von einer Ursache ausgehend die Wahrscheinlichkeit für das Eintreffen eines Ereignisses zu berechnen, wird für ein beobachtetes Ereignis die Wahrscheinlichkeit einer möglichen Ursache berechnet. Dies findet beispielsweise in der Medizin Anwendung, wo von diversen medizinischen Beobachtungen (in Form von Testergebnissen, Symptomen etc.) auf eine Krankheit (als Ursache) geschlossen wird. Formal wird für eine Abfragevariable X und eine Menge von Evidenzvariablen E mit den zugehörigen beobachteten Werten e sowie den verbleibenden unbeobachteten Variablen Y (mit ihren Werten y) die Abfrage $P(X|e)$ notiert. Die Auswertung lautet:

$$\begin{aligned} P(X|e) &= \alpha P(X, e) \\ &= \alpha \sum_{y} P(X, e, y) \end{aligned}$$

Die Umsetzung kann sowohl mit exakten Verfahren (Inferenz durch Aufzählung, Variablenelimination, Clustering; vgl. Russell & Norvig, 2004, Kap.14.4) als auch mit Näherungsverfahren (Direkte Sampling-Methoden, Markov-KettenSimulation; vgl. Russell & Norvig, 2004, Kap.14.5) erfolgen.

In Intelligenten Tutoriellen Systemen werden Bayes'sche Netzwerke verwendet, um beispielsweise Studentenmodelle zu realisieren. So wird in der Arbeit von Singley et al. (2000) ein Bayes'sches Netzwerk eingesetzt, das aus beobachtetem Verhalten auf verschiedene Rollen schließt (s. Kap.6.4.2).

Die eben vorgestellten Bayes'schen Netzwerke eignen sich als Modelle für das Schließen unter Unsicherheit in statischen Szenarien, das heißt jeder Zufallsvariablen ist ein fest definierter Wert zugeordnet. In der Praxis existieren hingegen

[1] engl. Mathematiker und Pfarrer, * 1702 n. Chr. in London, †17. April 1761 in Tunbridge Wells

auch Problemsituationen, in denen die dynamischen Aspekte eines Problems eine wichtige Rolle spielen. Der Grundgedanke bei der Beschreibung des Faktors *Zeit* besteht darin, den Änderungsprozess als eine Folge von Momentaufnahmen zu beschreiben, das heißt ein zeitlicher Verlauf wird in Zeitscheiben unterteilt. Jede Zeitscheibe t enthält eine Menge der zu diesem Zeitpunkt beobachtbaren Evidenzvariablen E_t und eine Menge nicht beobachtbarer Zustandsvariablen X_t. Ferner wird eine geeignete Beschreibung für den Übergang zwischen zwei Zeitscheiben t_i und t_{i+1} und die damit verbundenen Veränderungen benötigt. Prinzipiell kann jede Zeitscheibe unterschiedliche Variablen besitzen, der Einfachheit halber wird aber angenommen, dass die Variablen und ihre Verknüpfungen auf allen Zeitscheiben identisch sind, jede Zeitscheibe kann somit durch Replikation der vorherigen Zeitscheibe entstehen.

6.2.6.2 Dynamische Bayes'sche Netzwerke

Fasst man ein *„gewöhnliches"* Bayes'sches Netzwerk als eine wie eben beschriebene Zeitscheibe auf und ergänzt diese um geeignete Informationen bezüglich des Übergangs von einer Zeitscheibe zur nächsten, so erhält man ein dynamisches Bayes'sches Netzwerk (DBN). Damit wird es möglich, *„zeitlich veränderliche Aspekte der zu modellierenden Domäne explizit zu repräsentieren"* (Wittig, 2002, S.38). Die Verbindungen zwischen je zwei Zeitscheiben werden mittels sogenannter Übergangs-BWTs notiert, um somit die Veränderungen zwischen den Zeitscheiben zu modellieren. Insgesamt ergibt sich eine Struktur wie in der nachfolgenden Abbildung 6.4.

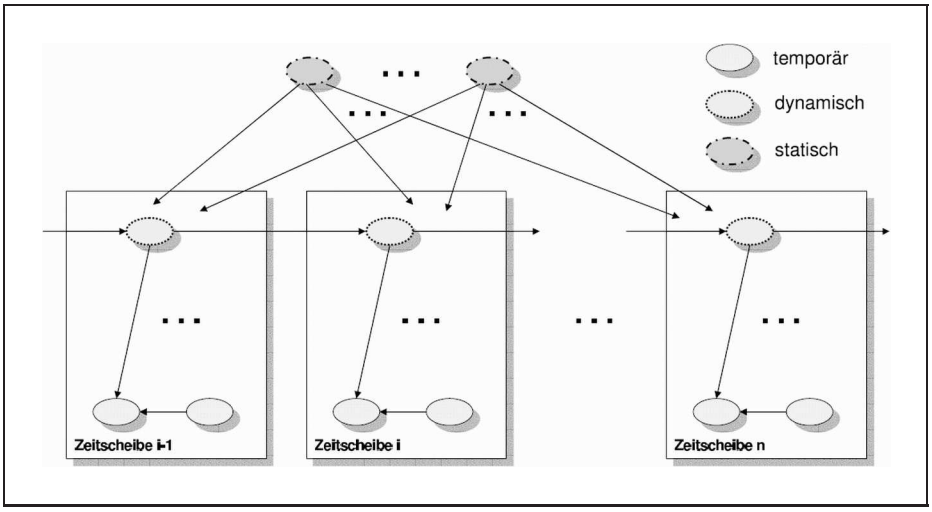

Abb. 6.4: Struktur eines Dynamischen Bayes'schen Netzwerks
(Wittig, 2002, S.38)

Die Knoten eines Dynamischen Bayes'schen Netzes werden wie folgt unterteilt:

- Temporäre Knoten sind nur innerhalb einer einzigen Zeitscheibe relevant. Entweder wird ihr Zustand als Evidenz betrachtet oder er wird im Rahmen

der Inferenz aufgrund von Einflüssen anderer Variablen berechnet. Sie haben keine direkte Auswirkung auf Knoten in anderen Netzwerken.

- Dynamische Knoten enthalten Variablen, deren Zustand sich im zeitlichen Verlauf ändern kann. Sie sind normalerweise in jeder Zeitscheibe mit gleichem Knotennamen enthalten und modellieren die dynamischen Aspekte des Bayes'schen Netzwerks und der zugehörigen Domäne.
- Statische Knoten enthalten Variablen, deren Zustand sich innerhalb des Dynamischen Bayes'schen Netzwerks beziehungsweise seiner Lebensdauer nicht ändert, sondern konstant bleibt. Der zugehörige Knoten wird im Netz außerhalb der Zeitscheiben notiert.

Interessante Anwendungsfälle von Dynamischen Bayes'schen Netzwerken – auch im Bereich von ITS – werden in (Wittig, 2002, Kap.2.6) vorgestellt.

6.2.6.3 Hidden-Markov-Modelle

Ein weiterer Ansatz, Prozesse zu modellieren, in denen sich die Werte von Zufallsvariablen über die Zeit hinweg ändern, stellen – neben weiteren Modellen – die Hidden-Markov-Modelle (HMMs) dar, benannt nach Andrei Andrejewitsch Markow[2]. Bei einem HMM handelt es sich um ein stochastisches Modell, in dem der Zustand des Prozesses mit einer einzigen diskreten Zufallsvariablen beschrieben wird. Genaugenommen stellen HMMs einen Spezialfall der zuvor erwähnten Dynamischen Bayes'schen Netze dar.

HMMs werden bevorzugt zur Erkennung von Mustern, von natürlicher Sprache und – in der Mensch-Maschine-Kommunikation – von Gesten eingesetzt. In ITS finden sie beispielsweise Anwendung bei der Erkennung und Klassifizierung von bestimmten Kommunikationssequenzen (s. Kap.6.4.4).

6.2.7 Software-Agenten

In den 1990er Jahren hat das Prinzip der Agenten Einzug gehalten in die Software-Entwicklung respektive in die Entwicklung intelligenter Software-Systeme (vgl. Russell & Norvig, 2004, S.49). Umgangssprachlich versteht man unter einem Agenten eine Art Stellvertreter, der im Auftrag einer anderen Person mit deren Vollmacht handelt (vgl. Johnson & Murch, 2000, S.17). Ein Agent konzentriert sich dabei in der Regel auf eine Aufgabe und nutzt dabei Fähigkeiten, über die sein Auftraggeber selbst nicht verfügt. Es muss sichergestellt sein, dass der Agent Zugang zu den Informationen besitzt, die zur Erledigung seiner Aufgabe notwendig sind (vgl. Johnson & Murch, 2000, S.21f).

Für software-basierte Agenten (oder Software-Agenten) existiert keine einheitliche, allgemein anerkannte Definition (vgl. Johnson & Murch, 2000, S.25ff; Kühnel,

[2] russ. Mathematiker und Pfarrer, * 14. Juni 1856 in Rjasan, †20. Juli 1922 in Petrograd

2001, S.203f; KLÜGL, 2001, S.13; FERBER, 2001, S.29ff; RUSSELL & NORVIG, 2004, S.55ff), vielmehr werden wesentliche Eigenschaften aus den unterschiedlichen Definition extrahiert: *„Wir erkennen Übereinstimmung darin, dass Agenten autonom, zielorientiert, beharrlich, logisch denkend, produktiv und kommunikativ sind"* (JOHNSON & MURCH, 2000, S.28). Mangels einer solchen Definition wird in dieser Arbeit ein Software-Agent betrachtet als *„ein Programm, das im Auftrag einer Person zielgerichtet und autonom handelt in Kommunikation innerhalb und mit seiner Umgebung"*.

Je nach Schwerpunkt von Betrachtungsweise, Einsatzgebiet oder realisierten Fähigkeiten werden Software-Agenten weiter differenziert. So spezifiziert man intelligente Agenten als solche Software-Agenten, die aufgrund ihrer Intelligenz in der Lage sind, sich an eine Umgebung anzupassen, zu lernen oder einen Vorgehensplan zur Erreichung ihrer Ziele zu entwickeln (vgl. JOHNSON & MURCH, 2000, Kap.2.4). Aufgrund dieser Anpassungsfähigkeit werden diese Agenten auch als adaptive oder lernende Agenten bezeichnet (vgl. RUSSELL & NORVIG, 2004, S.79ff). Mobile Agenten *„können sich auf einem Host- oder Client-Computer befinden und für die Ausführung ihrer Aufgaben durch andere Computer, Netzwerke oder das Internet wandern"* (JOHNSON & MURCH, 2000, S.68), wobei eine geeignete Infrastruktur vorausgesetzt werden muss. Für die Unterstützung von Lernenden in einem intelligenten Tutoring-System sind auch die Pädagogischen Agenten von Interesse, *„die im Sinne von Lernsystemen eingesetzt werden und Benutzer im Lernbereich anleiten und begleiten"* (BENDEL, 2003, S.72), deren Ziel also in der Unterstützung von Lernprozessen liegt. Speziell für Zwecke der Kommunikation eignen sich spezielle Agenten, die sogenannten *Chatterbots*: *„Chatterbots sind Agenten, die für das Chatten im Web verwendet werden"* (JOHNSON & MURCH, 2000, S.66).

Zwar stufen JOHNSON & MURCH (2000) diese Chatterbots als Form der Unterhaltung ein – und dies entspricht auch im Weitesten dem Charakter des allerersten dokumentierten Chatterbot namens *ELIZA* von Joseph Weizenbaum, der sein Programm nie als ernsthaften Gesprächspartner wissen wollte, sondern lediglich eine Demonstration des technisch Machbaren damit bezweckte (vgl. WEIZENBAUM, 1966) –, jedoch finden sich inzwischen auch ernsthafte Anwendungen dieser Technologie wie beispielsweise *Anna*[3], der Online-Assistent des Einrichtungshauses IKEA oder auch *Horst Förster*[4], der als virtueller Gesprächspartner des Forschungsprojekts *„Zukunftsorientierte Waldwirtschaft"* des Bundesministeriums für Bildung und Forschung umfangreiche Information zur nachhaltigen Waldwirtschaft in Form eines natürlichsprachlichen Dialog anbietet (s. Abb.6.5).

Zwischenzeitlich sind auch Software-Komponenten für den Einsatz von Chatterbots in eigenen Programmen wie beispielsweise *JAIMBot*[5] erhältlich. Auf architektonischer Ebene kann eine weitere Unterteilung in Einzelagentensysteme und Multiagentensysteme getroffen werden.

[3] Zu finden unter `http://www.ikea.com/ms/de_DE/local_home/homeshopping.html` - letzter Zugriff am 15.02.2007.

[4] Zu finden unter `http://www.zukunftswald.de/` - letzter Zugriff am 15.02.2007

[5] Zu finden unter `http://jaimbot.sourceforge.net/` - letzter Zugriff am 15.02.2007

Abb. 6.5: Beispiel eines Chatterbots
(`http://www.zukunftswald.de/`)

Ohne den nachfolgenden Darlegungen vorgreifen zu wollen, kann an dieser Stelle bereits auf die Bedeutung der soeben vorgestellten Konzepte der KI für den Einsatz in ITS hingewiesen werden: Erst die Möglichkeit, natürliche Intelligenz – wenn auch derzeit nur ansatzweise und in stark begrenzten Domänen – mittels Computersystemen nachzubilden, schafft die Voraussetzung für intelligente tutorielle Systeme, also für solche E-Learning-Angebote, die in der Lage sind, sich auf die Lernprozesse einzelner Lernender oder von Lerngruppen einzustellen.

6.3 Architektur intelligenter Tutor-Systeme

Der Aufbau eines ITS besteht üblicherweise aus Modellen für das Wissensgebiet, den Lerner und den Tutor sowie eine Kommunikationskomponente (vgl. Schulmeister, 1997; Puppe, 1992). Der Ablauf innerhalb einer solchen Architektur umfasst folgende Schritte:

1. Das System beobachtet den Lernenden über die Benutzerschnittstelle anhand seiner Interaktionen mit dem System. Diese Interaktionen beinhalten sämtliche während des Lernprozesses durchgeführte Bedienschritte und können auch als Lösungsversuch des Lernenden interpretiert werden. Aus diesen Beobachtungen generiert das System ein Modell des Lernenden und trifft in diesem Zuge Annahmen hinsichtlich seines Kenntnisstands und seiner Fähigkeiten.

2. Der Wissensstand des Lernenden wird – zusammen mit dessen Lösungsversuch – an das Tutormodell weitergereicht, wo im Bedarfsfall unter Zuhilfenahme des Wissensmodells ein angemessenes Hilfsangebot für den Lernenden generiert wird. Dazu muss das Tutormodell erkennen, wann eine Problemsituation (= Bedarfsfall) vorliegt und welcher Art das erkannte Problem ist.

3. Das Wissensmodell liefert dem Tutormodell eine Art Referenzmodell, das innerhalb des Tutormodells mit den Informationen des Lernermodells verglichen wird. Das Referenzmodell kann je nach Ausgestaltung des ITS ein Modell eines idealen Lerners enthalten oder aber auch Musterlösungen der zu bearbeitenden Aufgabe.

4. Anhand der Differenzen zwischen Lernermodell und Referenzmodell generiert das Tutormodell unter Berücksichtigung der zuvor identifizierten Problemsituation ein Unterstützungsangebot für den Lernenden und reicht dieses an die Benutzerschnittstelle weiter.

Die Abbildung 6.6 gibt die typische Architektur eines ITS inklusive der soeben skizzierten Vorgehensweise wieder.

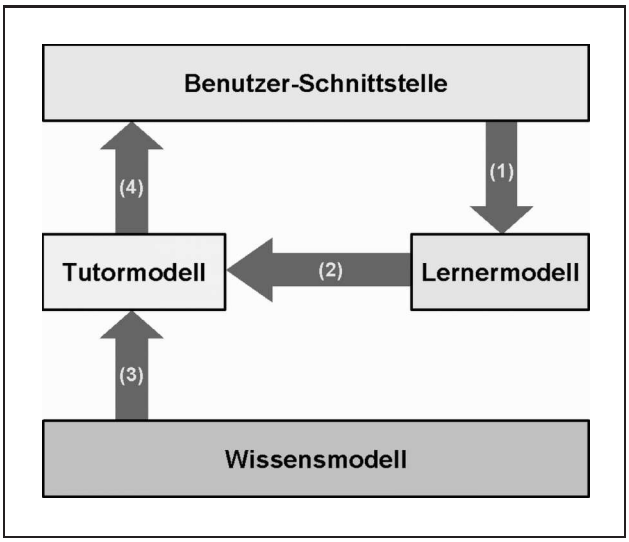

Abb. 6.6: Systemarchitektur eines ITS

Eine ähnliche Vorgehensweise findet sich beispielsweise bei Kerres 2001, S.71f, weitere – wenngleich nicht grundsätzlich verschiedene – Architekturansätze und Abläufe werden in Martens 2003, Kap.6 zusammenfassend vorgestellt.

6.3.1 Wissensmodell

Das Wissensmodell eines ITS enthält als Wissensbasis (s. Kap.6.2.1) das Wissen eines Experten hinsichtlich eines spezifischen Wissengebiets. Daher spricht man in diesem Zusammenhang auch oftmals vom Expertenmodell (vgl. Schulmeister, 1997, S.182). Neben der für Wissensbasen typischen Formulierung von explizitem

und deklarativem Wissen findet sich auch heuristisches Wissen in diesen Modellen wieder: *„Heuristisches Wissen besteht aus Erfahrungs- und Problemlösungswissen von Experten, das nicht an Inhalte gebunden ist; es besteht aus Handlungsempfehlungen für Transformationen, die der Lernende auf das Lernmaterial anwenden kann, oder für Operationen, mit denen Problemlösungsprozesse in handhabbare Aktivitäten unterteilt werden können"* (Schulmeister, 1997, S.182).

Die Umsetzung von Wissensmodellen reicht von eher schlichten Systemen, die auf Anfrage durch die Lernenden Problemlösungen anbieten, bis hin zu Expertensystemen (s. Kap.6.2.3), die nicht nur ihre Lösungsvorschläge erklären (vgl. Kerres, 2001, S.72), sondern darüber hinaus sogar in der Lage sind, *„dem Lernenden Vorschläge über die Reihenfolge des Lernstoffs zu unterbreiten"* (Schulmeister, 1997, S.183).

6.3.2 Lernermodell

Das Lernermodell eines ITS beschreibt die Abbildung des Lernenden auf eine Wissensbasis. Weil es die Grundlage für eine inhaltliche Diagnose der Lernsituation darstellt, wird das Lernermodell auch als Diagnosemodell bezeichnet (vgl. Schulmeister, 1997, S.183). Diese im System enthaltene Diagnosekomponente muss *„das Verhalten des Lernenden analysieren und Rückschlüsse über die Kompetenz des Lerners ziehen"* können (Kerres, 2001, S.71).

Zwar werden in der Literatur verschiedene Ansätze zur Ermittlung des Wissensstandes von Lernenden (wie das Differenzmodell oder das Fehlermodell) genannt (vgl. Schulmeister, 1997, S.183f; Kerres, 2001, S.72), letztlich jedoch muss sich ein Lernender in einem ITS anhand seiner Antworten (oder Handlungen) stets mit einem Experten beziehungsweise dessen im System hinterlegten, als optimal angenommenen Antworten (bzw. Handlungssequenzen) vergleichen lassen.

6.3.3 Tutormodell

Das Tutormodell entscheidet über Inhalt, Zeitpunkt und Form der Präsentation von Lernmaterialien (vgl. Kerres, 2001, S.71) und beschreibt folglich die Umsetzung der pädagogischen Strategien des ITS: *„Das Tutorenmodell simuliert das Entscheidungsverhalten eines Lehrers, den Entscheidungsprozeß, bezogen auf pädagogische Interventionen, und generiert angemessene Instruktionen, basierend auf den Differenzen zwischen Expertenmodell und Lernermodell"* (Schulmeister, 1997, S.186). Somit umfasst das Tutormodell pädagogisches Wissen und kann als Ergänzung zum Expertenwissen des Wissensmodell angesehen werden.

Der Eingriff in den Lernprozess durch das ITS kann anhand zweier unterschiedlicher pädagogischer Ansätze erfolgen (vgl. Schulmeister, 1997, S.186): dem *sokratischen Dialog* und dem *Coaching*.

6.3.3.1 Sokratischer Dialog

Die pädagogische Strategie des sokratischen Dialogs geht zurück auf Sokrates[6], der durch geschickte Dialogführung seinen Gesprächspartner zu Reflexionen über das soeben Gesagte zwang und ihm dabei zu neuen Erkenntnissen verhalf: *„Der Gesprächspartner erkannte sein Nichtwissen und erst der daraus resultierende Konflikt erweckte bei ihm die Neugierde und das Bedürfnis, in den Gesprächen mit Sokrates diesen Konflikt durch entsprechenden Erkenntnisgewinn aufzulösen. Dabei ging Sokrates davon aus, dass der Wissenszuwachs nicht von ihm ausgehen könne, sondern sich vielmehr aus der inneren Auseinandersetzung seines Gesprächspartners heraus entwickeln müsse. Ihm, Sokrates, käme lediglich eine Helferrolle zu"* (Hohenstein & Sander, 2005, S.8).

Diese Strategie wird beispielsweise von Stevens & Collins (1977) aufgegriffen und dem *Why System*, einem ITS zum Thema *Einflussfaktoren von Regenfällen*. Die Formalisierung der tutoriellen Strategie erfolgt mittels Produktionsregeln der Form *„If in situation X, do Y"* (Stevens & Collins, 1977, S.256), das Wissen selbst wird mit Skripten und Subskripten verfasst, die Sprachanalyse der Benutztereingaben beruht auf einer geeigneten Grammatik und versucht, die Antworten des Lernenden entsprechenden Schritten und Zuständen in den Skripten zuzuordnen. Weitere ITS, die das Konzept des sokratischen Dialog als pädagogische Strategie aufgreifen und umsetzen, werden beispielsweise in Park & Lee (2003), Lopez-Herrejon & Schulman (2004), Domeshek et al. (2004) sowie Wang et al. (2005) skizziert.

6.3.3.2 Coaching

Bei der pädagogischen Methode des Coaching werden in Problemsituationen typischerweise Tipps und Hinweise angeboten, von denen das System annimmt, dass diese zur erfolgreichen Bewältigung des jeweiligen Problems beitragen (vgl. Schulmeister, 1997, S.186). Dieser Ansatz fällt somit in den in Kapitel 5.7.2 bereits skizzierten Bereich des Tutoring und kann mit diesem synonym verwendet werden.

Während beim sokratischen Dialog versucht wird, dem Lernenden Erklärungen für Fehler zu entlocken (vgl. Schulmeister, 1997, S.187), bietet das Coaching (resp. Tutoring) dem Lernenden mehr oder weniger elaborierte Lösungen in Problemsituationen an. Folglich bietet sich die Methode des sokratischen Dialogs, bei der der wichtige Schritt des Erkenntnisgewinns beim Lernenden von innen heraus erfolgt, weniger an, wenn Wissensdefizite beim Lernenden ausgeglichen werden sollen.

Mittels eines simulierten Studenten (*simulated student*), der sich als (künstliches) Gruppenmitglied ausgibt, erforschen Vizcaíno & Du Boulay (2002) die Einsatzmöglichkeiten eines solchen Coachs in *HabiPro*, einem CICLS zur Einführung in die Programmierung: Je nach Problemsituation gibt der simulierte Student Hinweise und Erklärungen, motiviert zur Teilnahme oder moderiert die Gruppe während des Problemlösungsprozesses. Ähnliche Ansätze finden sich beispielsweise

[6] gr. Philosoph, * 469 v. Chr., †399 v. Chr. in Athen

auch in den Systemen *AlgebraJam* (vgl. Singley et al., 2000) oder *COLLAGEN* (vgl. Davies et al., 2001).

6.3.4 Kommunikationskomponente

Die Kommunikationskomponente (oder das *Interface* als Schnittstelle zwischen Anwender und System) gestaltet die Möglichkeiten zur Interaktionen zwischen Lerner und System mit dem Ziel, möglichst hohe Flexibilität und Adaptivität zu erreichen, und zwar sowohl im Hinblick auf den Lernprozess als auch auf den Lernenden selber (vgl. Schulmeister, 1997, S.187). Diese Interaktionen umfassen Befragungen und Moderationen des Lernenden durch das System wie auch Anfragen des Lernenden an das System beziehungsweise dessen Tutor, der sich ansonsten eher zurückhaltend gibt. Der Freiheitsgrad solcher Schnittstellen deckt ein breites Spektrum ab, das *„von völliger Kontrolle des Dialogstils bis zur völligen Freiheit des Lernens reicht"* (Schulmeister, 1997, S.188).

Eine möglichst hohe Unterstützung von Natürlichsprachlichkeit sowohl bei der Erkennung von Benutzereingaben als auch bei der Generierung von Ausgaben wird aber nicht unbedingt als wirklich relevant für effektives Tutoring erachtet. Oder wie es Schulmeister (1997) formuliert: *„Der natürlich-sprachliche Dialog ist ein möglicher Zusatz zu einem tutoriellen System, aber kein notwendiger"* (Schulmeister, 1997, S.189).

So lässt es sich erklären, dass trotz des inzwischen hohen Reifegrads bei der Erkennung natürlicher Sprache eine beachtliche Anzahl von ITS davon keinen Gebrauch macht, sondern auf strukturierte oder semi-strukturierte Kommunikationsschnittstellen ausweicht. Dabei wird dem Anwender über geeignete Oberflächenelemente eine begrenzte Auswahl an kommunikativen Items angeboten, die entweder ganze Sätze mit Standardphrasen wie beispielsweise *„Ja."*, *„Nein."*, *„Bist Du einverstanden?"* repräsentieren oder aber Satzanfänge anbieten, die vom Anwender nach Auswahl sinngemäß zu vervollständigen sind (s. Abb.3.1).

Üblicherweise erfolgt mit der Strukturierung von Kommunikation auch die Bildung einer Taxonomie kommunikativer Akte: Die Einordnung von Items in Primär- und Sekundärfunktionen mit unterschiedlichen semantischen Inhalten ermöglicht eine vergleichsweise einfache Analyse des Verlaufs von Gruppendiskussion während der Zusammenarbeit in kollaborativen Systemen (Baker & Lund, 1997, vgl.). Ähnliche Ansätze finden sich beispielsweise in den Systemen von Jermann (1999), Matessa (2001) oder auch Soller & Busetta (2003).

In intelligenten tutoriellen Systemen werden nicht notwendigerweise sämtliche der in diesem Kapitel vorgestellten Sub-Modelle vollständig umgesetzt. Vielmehr existiert eine Vielzahl von Systemen, die zu Forschungszwecken entwickelt und eingesetzt werden und eher als Prototypen denn als vollwertige Programme anzusehen sind (Schulmeister, 1997, vgl.).

Eine kleine Auswahl von solchen Systemen soll zum Abschluss dieser Heranführung an intelligente tutorielle Systeme – stellvertretend für die Vielzahl existierender

Ansätze, Prototypen und Systeme – skizziert werden.

6.4 Ausgewählte Tutor-Systeme

Allen nachfolgend präsentierten Tutor-Systemen gemeinsam ist ein Bezug in der einen oder anderen Weise zur vorliegenden Arbeit: Dieser Bezug kann sich in der Ausgestaltung der Tutorkomponente als virtuelles Teammitglied (HabiPro) äußern oder durch die Verwendung eines Rollenmodells (AlgebraJam) begründet sein oder aber durch eine spezielle Kommunikationsunterstützung (C-CHENE, EPSILON) gegeben sein.

6.4.1 HabiPro

Vizcaíno & Du Boulay (2002) entwickeln mit *HabiPro* ein ITS zur verteilten, kollaborativen, synchronen Programmierung mit dem Ziel, den Lernenden *gute Programmiergewohnheiten* zu vermitteln: *„HabiPro, from the Spanish «Habitos de programmación» (Programming Habits), is a collaborative, distributed, synchronous system designed to develop good programming habits in students"* (Vizcaíno & Du Boulay, 2002, S.3). In einer Lerneinheit wird den Teilnehmern ein fehlerbehaftetes Beispiel präsentiert, welches unter ausschließlicher Nutzung des HabiPro-Systems (s. Abb.6.7) korrigiert werden muss.

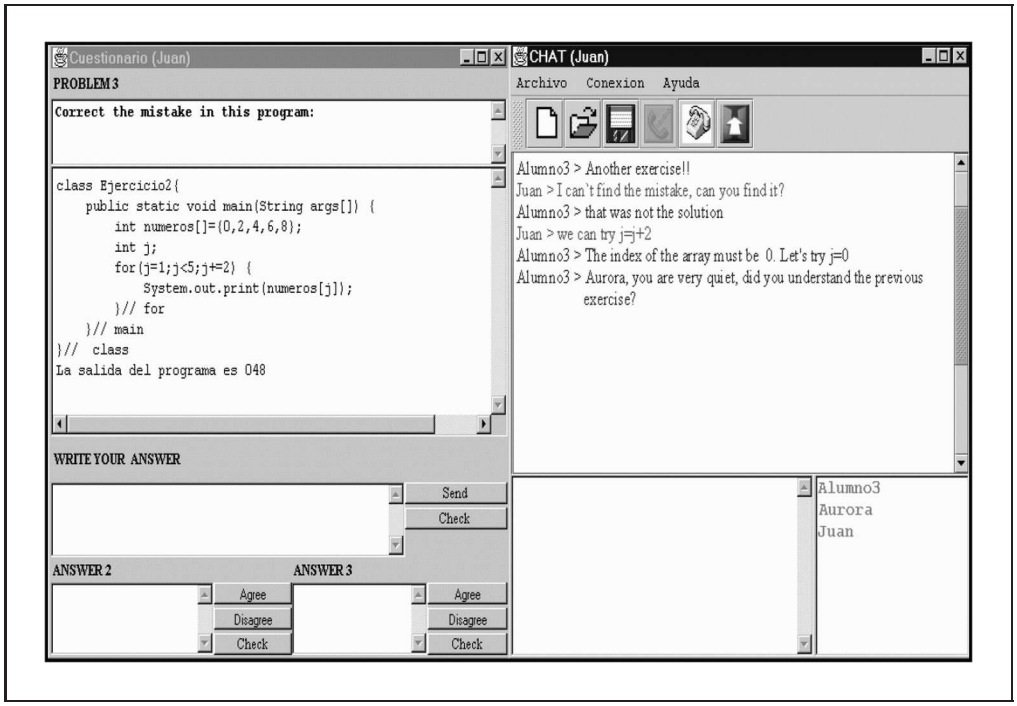

Abb. 6.7: Programmieren lernen mit HabiPro
(Vizcaíno & Du Boulay, 2002, S.4)

Das Kernstück von HabiPro bildet eine tutorielle Komponente, die in Form eines künstlichen Teammitglieds, des *Simulated Student*, in Erscheinung tritt, den Lernprozess beobachtet und bei Bedarf einschreitet. Dabei kommen je nach erkannter Problemsituation unterschiedliche Strategien zur Anwendung (vgl. Vizcaíno & Du Boulay, 2002, S.2).

Das Modell dieses *„simulierten Studenten"* setzt sich zusammen aus einer Menge individueller Lernermodelle (*Student Model*; SM), aus einem Gruppenmodell (*Group Model*; GM) und aus dem *„Simulated Student Behaviour Model"* (SSBM), welches das eigentliche Tutormodell darstellt: *„The SSBM uses the information stored in the Group Model and in the Student Models to decide when and how the Simulated Student has to intervene"* (Vizcaíno-Barceló, 2002, S.2).

Ergänzende Komponenten sind ein Interface als Schnittstelle zwischen System und Studenten sowie ein *Information Manager*, der sämtliche eingehende Informationen aus dem Interface klassifiziert und an nachfolgende Komponenten weiterreicht. Die Gesamtarchitektur von HabiPro ist in Abbildung 6.8 skizziert.

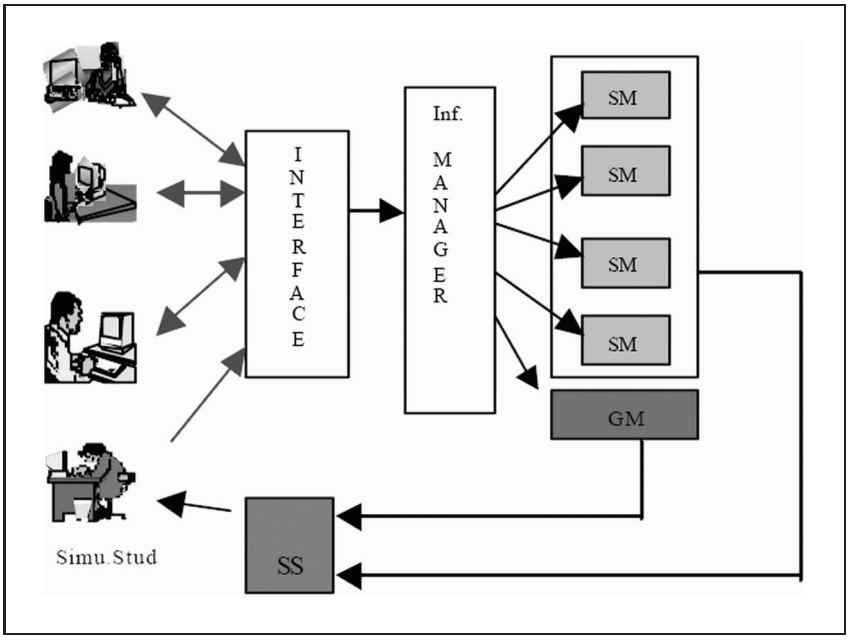

Abb. 6.8: Architektur von HabiPro
(Vizcaíno-Barceló, 2002, S.2)

Prinzipiell ist der Simulated Student in der Lage, in ausgewählten Problemsituationen einzugreifen. Dazu bedient er sich diverser Strategien (Hinweise geben, konkrete Beispiele liefern, nachfragen etc.; vgl. Vizcaíno & Du Boulay, 2002, S.2), die in einem pädagogischen Modell hinterlegt sind. Obwohl das ITS in wenigen Fällen fälschlicherweise zu oft oder gar nicht eingriff, konnten Vizcaíno & Du Boulay (2002) nachweisen, dass die Problemlösungsrate unter den Teilnehmern, bei denen der Tutor zum Einsatz kam, höher war als bei denjenigen Teilnehmern, die ohne Tutor arbeiteten.

6.4.2 AlgebraJam

AlgebraJam von SINGLEY ET AL. (2000) ist ein ITS, das die synchrone, gemeinsame Bearbeitung von Problemstellungen aus dem Gebiet der elementaren Algebra unterstützt. Die Oberfläche des Systems ist unterteilt in verschiedene Bereiche (s. Abb.6.9). Die zu bearbeitende Aufgabe, die in der linken unteren Ecke beschrieben wird, ist stets in einen fiktiven, aber realitätsnahen Kontext eingebettet und bewusst vage und unvollständig formuliert. Zum Auffinden fehlender Informationen wird ein fiktives Bücherregal (mit Telefonbuch, Stadtplan und weiteren relevanten Büchern) angeboten. Durch die Kombination dieser beiden Elemente wird nicht nur die Forderung nach Situiertheit erfüllt (s. Kap.5.4.3), sondern darüber hinaus auch das explorative Lernen unterstützt.

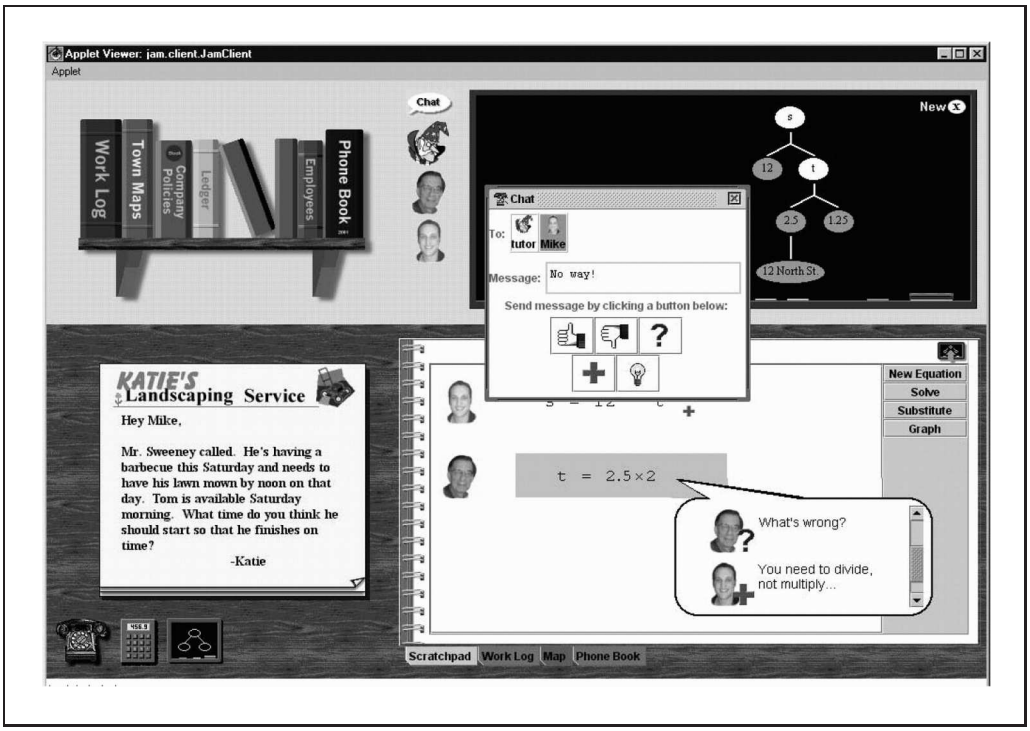

Abb. 6.9: Die Oberfläche von AlgebraJam
(SINGLEY ET AL., 2000, S.4)

Den gemeinsamen Arbeitsbereich bildet das sogenannte *Team Blackboard*, in welchem die bisherigen Arbeitsergebnisse graphisch repräsentiert werden: *„The team blackboard ... is a specialized representation that conveys the underlying logical structure of the problem. It is overloaded with function and represents perhaps the primary means for establishing and maintaining common ground in Algebra Jam"* (SINGLEY ET AL., 2000, S.149).

Ein persönliches Notizbuch bietet dem Einzelnen schnellen Zugriff auf ausgewählte Ressourcen sowie die Möglichkeit, individuelle Nebenrechnungen durchzuführen. Die Kommunikation zwischen den Teilnehmern erfolgt mittels text-basiertem Chat und sogenannten *Collabicons*: *„To further reduce the generation and comprehesion loads on participants, we are attempting to define a message typology which when*

completed will constitute a basic ontology of collaborative acts. The message types, which we have dubbed collabicons*, provide the sender wiht a small but fairly comprehensive range of collaborative intentions with which to frame a message"* (Singley et al., 2000, S.150). Somit steht den Anwendern von AlgebraJam eine Kommunikationskomponente zur Verfügung, die den Ansatz der strukturierten Kommunikation verfolgt. Die in Singley et al. (2000) beschriebene Topologie kommunikativer Akte besteht aus fünf Einträgen (Zustimmung, Ablehnung, Hilfegesuch, Hilfeangebot, Ideen-/Meinungsangebot), die um geeignete Textbotschaften erweitert werden können. Dieser Ansatz unterstützt damit minimale Anforderungen hinsichtlich aufgabenorientierter Kommunikation zur Problemlösung und ermöglicht gleichzeitig eine Vereinfachung sowohl bei der Erzeugung als auch bei der Analyse von Botschaften (vgl. Singley et al., 2000, S.150).

Die Tutorenkomponente kann bei Bedarf aktiviert werden und agiert dann als virtuelles Teammitglied (*Virtual Participant*). Die Umsetzung erfolgt als *Tutoring Agent* unter Zuhilfenahme der in Kapitel 6.2.7 skizzierten Agententechnologie und bedient sich derselben Schnittstelle wie die übrigen Teilnehmer: *„In our system architecture, the tutoring agent operates much like any other client and «collaborates» by sending the same kinds of events and making use of the same interface machinery as any other participant. The tutor can point at objects, send chat messages, critique and/or commend others' work, expose portions of the goal tree in the team blackboard, and even do productive work in the shared workspace in tethered mode"* (Singley et al., 2000, S.151).

Wie auch in HabiPro (s. Kap.6.4.1) kommt in der tutoriellen Komponente von AlgebraJam ein Gruppenmodell zur Anwendung, dem wiederum ein Rollenmodell zugrunde liegt: nach der Theorie von Vygotsky (1978) erfolgt der Wissenserwerb schrittweise und unter Einbeziehung eines erfahreneren Partners (*Zone of proximal development*). Singley et al. (2000) leiten daraus fünf Rollen ab, die für das kollaborative Lernen in AlgebraJam relevant und hilfreich sind: *„Based on a very broad reading of current social learning theorist ..., we are proposing a set of collaborative roles that students tanke on as they acquire cognitive skills in group settings. In any particular problem-solving episode, a participant may fill out each of these roles to a greater or lesser extent, but our position is that there is pedagogical value in having each participant assume these different points of view as learning progresses"* (Singley et al., 2000, S.151). Den einzelnen Rollen werden dabei verschiedene Aufgaben zugeordnet (s. Tab.6.1).

Tab. 6.1: Merkmale kollaborativer Rollen

Rolle	**Merkmale**
Beobachter (Observer)	Beobachtet Problemlösungsmaßnahmen der anderen Mitglieder; stellt klärende Fragen; initiiert Hilfegesuche; erhält Hilfe
Lehrling (Apprentice)	Liefert Werte für routinemäßige Teilziele; erhält Hilfe

Fortsetzung auf nächster Seite

Tab. 6.1: Merkmale kollaborativer Rollen *Forts.*

Rolle	**Merkmale**
Spezialist (Specialist)	Liefert Werte für nicht-routinemäßige Teilziele
Anführer (Leader)	Veröffentlicht Teilziele im Team Blackboard; weist Teilziele im Team Blackboard zu; identifiziert geeignete Ressourcen; bewertet Arbeitsprodukte
Trainer (Coach)	Antwortet auf Hilfegesuche anderer Teilnehmer; bietet Hilfe an; bewertet Arbeitsprodukte; modelliert korrektes Verhalten

(vgl. Singley et al., 2000, S.152)

Das System kann nun entweder den Teilnehmern Rollen zuweisen (*prescriptive mode*) oder aber versuchen, die von den Teilnehmern eingenommenen Rollen zu identifizieren (*non-prescriptive mode*). Im erstgenannten Modus werden die Kommunikationsmöglichkeiten im Interface gemäß Tabelle 6.1 eingeschränkt, was dem System aber die Analyse der Zusammenarbeit und ein mögliches korrigierendes Eingreifen erleichtert. Im zweiten, eher liberalen Arbeitsmodus versucht das System anhand der Benutzeraktionen die eingenommenen Rollen zu erkennen. Einen Lösungsansatz, um der diesem Problem anhaftenden Unsicherheit zu begegnen, sehen die Autoren in der Verwendung eines Bayes'schen Inferenz-Netzwerks (s. Kap.6.2.6).

Auch wenn die Arbeiten an AlgebraJam – nicht zuletzt aufgrund der Komplexität, die in kollaborativen Lernsituationen beobachtet werden kann – in 2001 enden, sehen die Autoren ihr System als einen (noch zu verfeinernden) Prototypen, der positiven Einfluss auf zukünftige ITS ausüben wird: *„Our working definitions of collaborative roles will be refined with experience with the system, and we hope to discover which groupings yield the most productive collaborations. More importantly, we hope that systems like Algebra Jam will create a new type of learning experience that harnesses the power of collaboration to motivate, accelerate, and deepen learning"* (Singley et al., 2000, S.151). Zumindest die Idee von Singley et al., einen rollenbasierten Unterstützungsansatz zu eruieren, hat ihren Weg in die vorliegende Arbeit gefunden.

6.4.3 C-CHENE

Bei C-CHENE (*Collaborative CHaine ENErgetique*; dt. kollaborative Energieketten) von Baker & Lund (1996) handelt es sich um *„a CSCL environment for co-construction of the concept of energy in physics"* (Baker et al., 2001, S.90). Studierende bearbeiten in Zweiergruppen einfache Aufgaben aus der Physik beziehungsweise der Elektrodynamik: *„The students' task was to draw an energy chain that represented a battery connected by two wires to a lighted bulb. Each student had a handout describing elements of the energy model. The task was a modeling problem and involved establishing relations ... between objects and events (e.g.*

battery, bulb, electricity), to concepts of the energy model and theory (reservoirs, transformers and transfers of energy)" (Lund, 1999, S.149). Dabei steht den Teilnehmern innerhalb der Oberfläche der CSCL-Umgebung C-CHENE (s. Abb.6.10) ein Bereich zur gemeinsamen, graphisch-basierten Konstruktion von Energieketten – bestehend aus Batterien, Glühbirnen, Drähten etc. – zur Verfügung, der ergänzt wird um eine text-basierte Kommunikationskomponente für die Koordination der zu erledigenden Teilaufgaben.

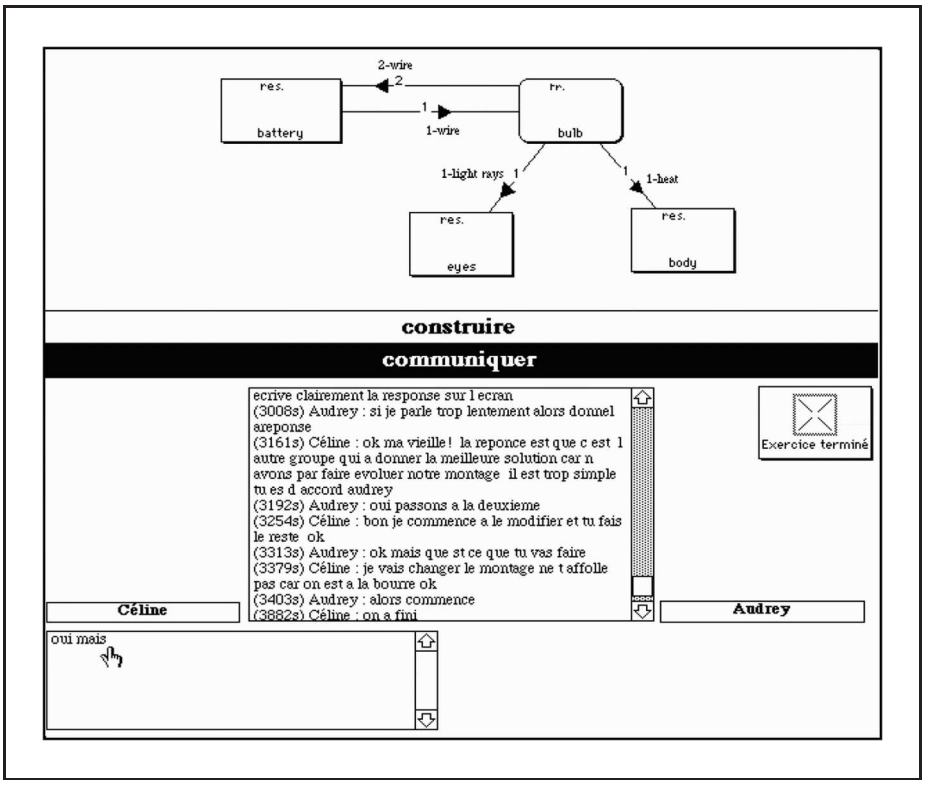

Abb. 6.10: Die Oberfläche von C-CHENE
(Baker & Lund, 1997, S.181)

Um zu untersuchen, wie Interaktionen zwischen den Lerndenden durch eine tutorielle Komponenten gefördert werden können, kann die Kommunikation auf zwei unterschiedlichen Arten stattfinden: neben einem dialogbasierten Chat für das Verfassen und den Austausch von freien Textbotschaften existiert auch eine strukturierte Kommunikationsschnittstelle (s. Kap.3.1.4, Abb.3.1). Sämtliche Kommunikation wird (für spätere Auswertungen) in Protokolldateien gespeichert.

Der in Abbildung 3.1 (s. Kap.3.1.4) dargestellte Dialog ist unterteilt in Konstruktions- und Kommunikationsbereiche. Die enthaltenen Schaltflächen senden nach Betätigung – je nach Typ – eine vordefinierte Botschaft ab (wie *„Ich schlage vor ..."*), bieten – im Falle von konstruktionsbezogenen Aufgaben – weitere Auswahlmöglichkeiten an (wie bspw. *„Erzeugung einer Energiequelle"*) oder führen zu einem vorgegebenen Satzanfang, der in einem kleinen Dialogfenster durch den Lernenden vervollständigt wird, bevor der auf diese Weise komplettierte Satz als Botschaft abgeschickt wird.

In einer empirischen Studie mussten sich die Teilnehmer zu Paaren zusammenfinden und unter ausschließlicher Nutzung von C-CHENE geeignete Aufgaben bearbeiten. Jede Gruppensitzung war auf drei Stunden begrenzt. Die eine Hälfte der Teilnehmer (vier Paare) arbeiteten mit dem dialogbasierten Chat, die andere Hälfte verwendete die strukturierte Kommunikationsschnittstelle. Nach Auswertung aller Protokolle schließen Baker et al., dass die Gestaltung der Kommunikationskomponente merklichen Einfluß auf die Qualität der Interaktionen zwischen den Teilnehmern während einer Lernsitzung ausübt. Insbesondere hat sich die Verwendung der strukturierten Kommunikationsschnittstelle (s. Abb.3.1) als förderlich für die Interaktionen gezeigt: *„providing the right degree of constraint on typewritten CMC can in fact promote an interaction more focussed on reflexion and the fundamental concepts at stake"* (Baker et al., 2001, S.91).

Eine Garantie für die ausschließliche Fokussierung auf die zu bearbeitende Aufgabe stellt die Verwendung einer strukturierten Kommunikationsschnittstelle jedoch keineswegs dar. Trotzdem bietet dieser Ansatz gewisse Vorteile im Hinblick auf die Auswertbarkeit von Gruppensitzungen, auf die später noch genauer eingegangen wird (s. Kap.8.4.3 und Kap.10).

6.4.4 EPSILON

Im Kontext des Projekts EPSILON (*Encouraging Positive Social Interaction while Learning On-Line*) befassen sich Soller et al. mit kollaborativen Lernszenarien und dort speziell mit sozialen Interaktionen, die zum Austausch von Wissen unter den Beteiligten führen. Die Untersuchungen zum Wissensaustausch in der Gruppe werden unter Zuhilfenahme des gleichnamigen Systems in der Domäne objektorientierter Modellierung nach OMT (*Object Modeling Technique*; vgl. Rumbaugh et al., 1991; Rumbaugh, 1996) durchgeführt: Zwölf Gruppen mit je drei Mitgliedern erhalten zunächst eine halbstündige Einleitung in das Themengebiet OMT, gefolgt von einer ebenso langen Einführung in die Bedienung der EPSILON-Software (s. Abb.6.11). Anschließend werden die Teilnehmer einer Gruppe auf unterschiedliche Räume verteilt, wo jeder Teilnehmer mit zusätzlichem individuellen Wissen hinsichtlich spezieller OMT-Konzepte instruiert wird, welches in einem kurzen Test überprüft wird.

In der nachfolgenden Online-Sitzung (mit einer Dauer von ca. 75 Minuten) bearbeiten die Teilnehmer gemeinsam eine Aufgabe, deren Ziel in der objektorientierten Modellierung eines gegebenen Sachverhalts liegt (wie bspw. dem Aufzeigen aller Relationen, die zwischen den Objektklassen *Angestellter*, *Firma*, *Bibliothek*, *Arbeitsvertrag*, *Computer* und *Zubehör* existieren). Während dieser Sitzung werden sämtliche Aktionen der Teilnehmer (inkl. Konversation) vom System aufgezeichnet, so dass die Sitzungsdaten nach Abschluss aller Sitzungen hinsichtlich diverser Fragestellungen analysiert und ausgewertet werden können (vgl. Soller, 2004, S.356ff).

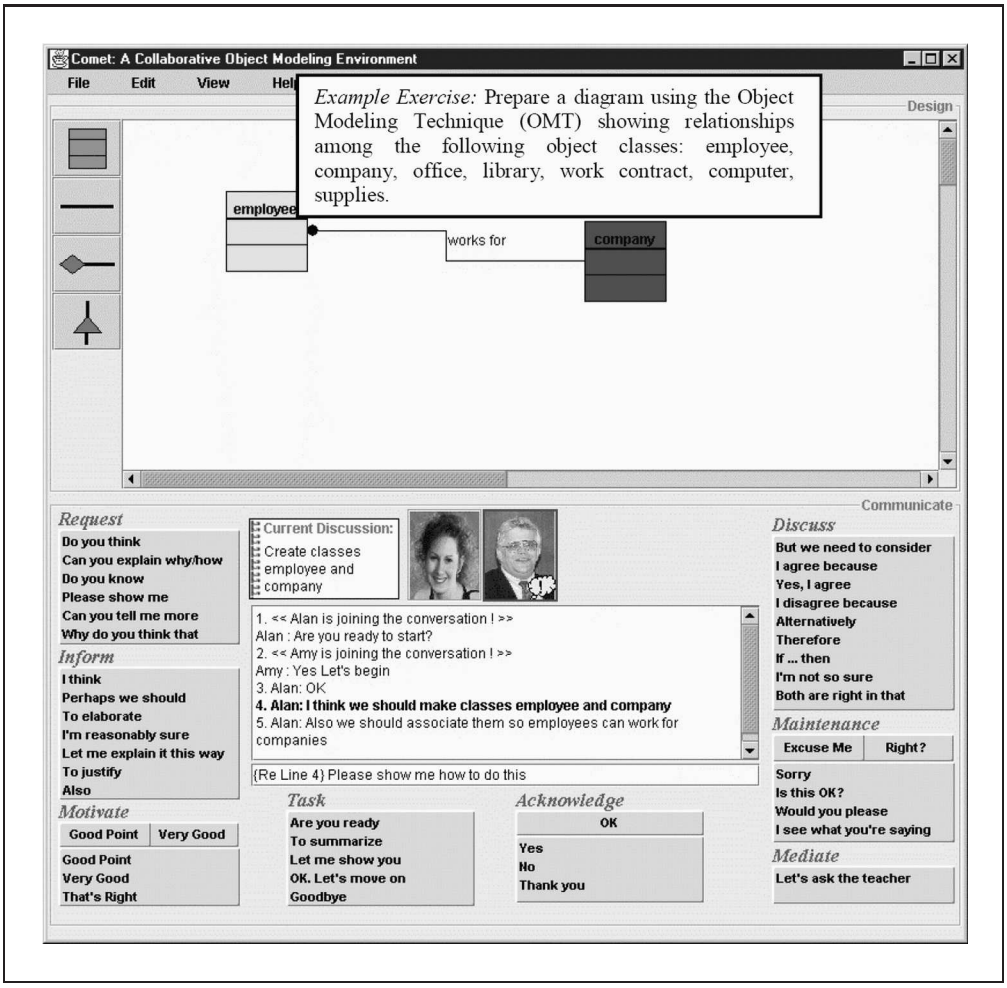

Abb. 6.11: Die Oberfläche von EPSILON
(Soller & Busetta, 2003, S.4)

Die Oberfläche der EPSILON-Applikation (s. Abb.6.11) präsentiert sich dem Anwender zweigeteilt:

- Die obere Hälfte enthält einen Arbeitsbereich, den *Shared OMT Workspace*, in welchem alle Werkzeuge bereitgestellt werden, die für die Konstruktion von objektorientierten Modellen mittels OMT als notwendig erachtet werden, wie beispielsweise Symbole für Klassen und für Assoziationen zwischen Klassen.
- In der unteren Hälfte befindet sich die Kommunikationskomponente von EPSILON, bei der es sich um eine strukturierte Dialogschnittstelle auf Basis der *Collaborative Learning Skills* nach McManus & Aiken (1995) handelt (s. Kap.3.1.4). Jeder Schaltfläche dieser Kommunikationskomponente ist entweder ein kompletter Satz (*„Yes."*, *„No."*, *„That's right."* etc.) oder ein Satzanfang (wie bspw. *„Perhaps we should ..."*), der vom Anwender zu vervollständigen ist (s. Kap.6.4.3), zugeordnet.

Insgesamt können die positiven Effekte strukturierter Dialogschnittstellen wie beispielsweise eine stärkere Fokussierung auf die eigentliche Aufgabe bestätigt werden.

Darüber hinaus ergibt sich aus Sicht des Systems eine einfachere Möglichkeit der Kommunikationsanalyse: jedem Satz (bzw. Satzanfang) der Dialogschnittstelle ist einerseits eine bestimmte konversationale Absicht zugeordnet, andererseits kann auf einen Parser zur Erkennung natürlicher Sprache verzichtet werden, da sich aus den Elementen der Dialogschnittstelle ein einfacher Code ableiten lässt.

Um nun die Frage zu klären, wie innerhalb eines Lernprozesses effektiv Wissen ausgetauscht wird, werden aus einzelnen Sitzungsprotokollen manuell diejenigen Segmente extrahiert, in denen die Teilnehmer ihr jeweiliges Individualwissen austauschen. Nach einer Klassifizierung dieser Segmente in erfolgreiche und erfolgose *knowledge sharing episodes* (vgl. Soller, 2004, S.357) wird eine KI-Komponente (genaugenommen ein verborgenes Markov-Modell; s. Kap.6.2.6)mit diesen Segmenten trainiert, so dass ein ITS zukünftig selber solche Phasen des Wissensaustauschs erkennen und bewerten kann.

Die Untersuchungen von Soller et al. zeigen, dass auf diese Weise Einbrüche in Lernphasen nicht nur erkannt, sondern auch erklärt werden können: *„This research explored the notion that HMMs might be used to identify and explain when and why students have breakdowns during knowledge sharing conversations"* (Soller, 2004, S.373). Darauf aufbauend sind geeignete tutorielle Strategien (vgl. Soller & Lesgold, 1999, S.65) umzusetzen, so dass das ITS EPSILON in Problemsituationen geeignete tutorielle Unterstützung anbieten kann.

Insgesamt stellen CSCL-Systeme – und insbesondere auch ICLS mit einer zusätzlichen tutoriellen Komponente – einen zeitgemäßen Ansatz dar, Unterstützung bei der Durchführung von gruppenbasierten Lernszenarien anzubieten. Die Vielzahl von Projekten, die eben diese Systeme als Forschungsgegenstand betrachten, belegt dies und zeigt auf, dass Forschung in diesem Gebiet gleichsam interessant wie notwendig ist. Trotz der Kritik, der sich solche Systeme immer wieder stellen müssen (vgl. Schulmeister, 1997, S.184f; Wessner, 2001, S.195), sind die erreichten Ergebnisse hinsichtlich der Unterstützung von Gruppen in Lernsituationen vielversprechend (vgl. Vizcaino, 2005; Rickel et al., 2002).

Ob diese Ergebnisse ausschließlich der KI und ihren Konzepten zuzuschreiben sind, darf nach Ansicht des Autors in Frage gestellt werden, schließlich hat Weizenbaum in den 1960ern mit ELIZA bereits eine geschickte Umsetzung eines sokratischen Dialogs erreicht, indem er sein Programm nach Schlüsselwörtern parsen ließ und entsprechende Kommunikationsbeiträge generiert hat, mit denen der menschliche Gegenpart zur Fortführung der Kommunikation animiert werden sollte.Diese Technik findet sich auch heute noch in einigen Chat-Bots wieder.

Die hier vorgestellten Systeme können aufgrund der Vielzahl solcher Systeme lediglich eine stark begrenzte, aber für die vorliegende Arbeit relevante Auswahl darstellen. Deren gemeinsames Ziel besteht in der Lerngruppenunterstützung und findet in unterschiedlichen Domänen Anwendung wie beispielsweise der Java-Programmierung (s. Kap.6.4.1), der objektorientierten Modellierung (s. Kap.6.4.4) oder der Elektrotechnik (s. Kap.6.4.3), um nur einige Themengebiete zu nennen. In diesem Zusammenhang kommen zur Erreichung des genannten Ziels unterschiedlichste Konzepte und Vorgehensweisen zum Einsatz. Die vorliegende Arbeit hat

sich zum Ziel gesetzt, dieses Forschungsgebiet und die aus ihm resultierenden Projekte und Lernsysteme durch die Kombination verschiedener Konzepte im Projekt *VitaminL* zu bereichern.

Kapitel 7

Programmieren lernen

Beim Erlernen einer Programmiersprache – beispielsweise im Rahmen eines informationswissenschaftlichen oder informatiknahen Studiums – erweist sich Gruppenrespektive Teamarbeit als durchaus vorteilhaft: Es bringen sich nicht nur mehrere Teilnehmer mit zum Teil sehr unterschiedlichen Fähigkeiten und Fertigkeiten in den Lernprozess ein, darüber hinaus können oftmals selbst kleinere Projekte erst in der Gruppe sinnvoll bearbeitet und erfolgreich umgesetzt werden. Gründe dafür liegen unter anderem in der Vielzahl von Einzelanweisungen, die meist für selbst einfache Algorithmen und Programmteile notwendig sind, sowie in den kurzen Zeitspannen, die – bedingt durch die oftmals auf ein Semester begrenzte Dauer einer Programmierveranstaltung – für die Bearbeitung der einzelnen Aufgaben jeweils zur Verfügung stehen.

7.1 Traditionelle Unterrichtsform

Das Erlernen von Elementen und Konzepten einer Programmiersprache sowie deren zielgerichtete Verwendung zur Formulierung von Problemlösungen (in Form von Algorithmen) stellt sich in seiner ursprünglichen Form als eine Art mehrstufiger Prozess dar, bei dem sich Phasen der Theorievermittlung und solche mit praktisch-orientierten Einheiten abwechseln.

7.1.1 Vermittlung theoretischer Inhalte

Die Vermittlung der theoretischen Grundlagen einer Programmiersprache wie Java wird üblicherweise als Vorlesung durchgeführt. In Einheiten von üblicherweise 90 Minuten präsentiert der Dozent per Frontalunterricht die jeweiligen Lehrinhalte mit steigendem Schwierigkeitsgrad beziehungsweise wachsender Komplexität. In dieser Unterrichtssituation findet üblicherweise eine einseitige 1-n-Kommunikation zwischen dem Dozenten und seinem Auditorium statt, in welcher Fragen an den Dozenten eher die Ausnahme sind. Die erste Vorlesung einer solchen Veranstaltung

beinhaltet beispielsweise eine Einleitung, in welcher die Studierenden an die Programmierung herangeführt werden, und ein allererstes Beispiel in der jeweiligen Programmiersprache, wobei traditionell die Ausgabe *„Hello, world!"* (oder auch *„Hallo, Welt!"*) auf dem Bildschirm eines Rechners ausgegeben wird.

In nachfolgenden Vorlesungseinheiten werden zunächst die Grundelemente der Programmiersprache wie Klammer- und Punktzeichen, Kommentare, erste Schlüsselwörter sowie einfache Anweisungen (Datendefinitionen, Ein- und Ausgabebefehle sowie die Bildung von Ausdrücken) behandelt, gefolgt von den Kontrollstrukturen (Sequenz, Auswahlen, Wiederholungen, Unterprogramme). Durch entsprechende Kombination dieser Sprachelemente können bereits nach wenigen Wochen Algorithmen zum Suchen und Sortieren von Daten formuliert werden.

Im Fall der Programmiersprache Java können anschließend weiterführende Konzepte wie beispielsweise die Objektorientierung und Möglichkeiten hinsichtlich deren Verwendung thematisiert werden. In dieses Themengebiet fallen sowohl Modellierungen einfacher Sachverhalte mittels Klassen und Objekten als auch die Gestaltung und Erstellung von GUI-Programmen unter Zuhilfenahme entsprechender Java-Klassenbibliotheken (AWT bzw. JFC/Swing). Eine Abrundung erfährt solch eine Vorlesung in der Regel durch spezielle Themen wie die Ein-/Ausgabe beliebiger Daten mit den sogenannten *Streams* (dt.: Datenströme) oder auch die Fehlerbehandlung mit dem Konzept der *Exceptions* (dt.: Ausnahmen).

Das Ziel der Veranstaltung besteht darin, den Teilnehmer nicht nur wichtige Sprachelemente und deren Bedeutung sowie grundlegende Konzepte der verwendeten Programmiersprache zu vermitteln, sondern darüber hinaus die Teilnehmer auch zu deren effektiver Anwendung zu befähigen. Am Ende der Veranstaltung sollten die Teilnehmer in der Lage sein, diese Befähigung anhand eines verhandelbaren Abschlussprojekts zu demonstrieren.

7.1.2 Erlangung praktischer Fertigkeiten

Die praktische Anwendung theoretischer Kenntnisse ist integraler Bestandteil von Programmierveranstaltungen jeglicher Art, sei es in Schulen, in Universitäten oder in sonstigen Bildungseinrichtungen. Schließlich liegt deren Ziel ja gerade darin, eine Fertigkeit zu vermitteln, die neben der Fähigkeit, bestimmte Probleme zu lösen, auch die Umsetzung solch einer Problemlösung mittels der jeweiligen Programmiersprache beinhaltet. So ist es naheliegend, die bereits erwähnte Theorievermittlung um praxis-orientierte Einheiten zur Festigung der Theorie zu ergänzen.

In der Praxis hat sich ein mehrstufiges Szenario bewährt, das verschiedene, aufeinander aufbauende Formen praktischer Übungseinheiten von einfachen Beispielen über Übungsaufgaben bis hin zu komplexen Projekten umfasst. Auf diese Weise werden die Studierenden sukzessive mit steigendem Schwierigkeitsgrad an die Lerninhalte herangeführt.

Allen praktischen Übungseinheiten gemeinsam ist die Durchführung an einem geeigneten PC-Arbeitsplatz, welcher es dem Lernenden gestattet, Quelltexte zu er-

stellen, mittels Java-Compiler zu übersetzen und den resultierenden Objekt- oder Ziel-Code (den sogenannten *Java-Byte-Code*) dem Java-Interpreter zur Ausführung zu übergeben.

7.1.2.1 Beispiele

Beispiele sind integraler Bestandteil der Lehrinhalte und dienen der allerersten praktischen Demonstration des jeweiligen zu vermittelnden Stoffes. Der Beispieltext wird zunächst analog zu den übrigen Lehrinhalten dem Auditorium präsentiert. Je nach Fortschritt in der Vorlesung und entsprechendem Umfang handelt es sich um einen vollständigen, lauffähigen Quelltext oder aber um für das jeweils behandelte Thema relevante Ausschnitte aus einem Quelltext.

Nach einigen Erläuterungen erfolgt dann üblicherweise die Ausführung des zum Quellcode gehörenden Objekt-Codes mit dem Ziel, den Effekt des zugrundeliegenden Quelltexts praktisch zu demonstrieren. Nachfolgende Bemerkungen, in Ausnahmefällen auch kurze Diskussionen, schließen solch ein Beispiel ab.

7.1.2.2 Übungsaufgaben

Unter einer Übungsaufgabe wird eine praktische Übungseinheit verstanden, die unmittelbar an die Theorievermittlung folgt und zeitlich wie organisatorisch in die zugehörige Vorlesungseinheit integriert ist. Sie bietet den Lernenden Gelegenheit, ausgewählte Aspekte der zugehörigen Vorlesungseinheit erstmalig praktisch umzusetzen. Notwendige Voraussetzung für diese Form von Übungseinheit ist das Vorhandensein von geeigneten Arbeitsplätzen, das heißt die Integration von Übungsaufgaben in die Vorlesungseinheiten ist nur dann sinnvoll durchführbar, wenn diese beispielsweise in einem Rechnerpool stattfindet.

Entsprechend der Einbettung in eine Vorlesungseinheit fallen der Umfang der Aufgabenstellung sowie die den Teilnehmern zur Verfügung stehende Bearbeitungszeit aus: Der übliche zeitliche Rahmen für die komplette Bearbeitung einer Übungsaufgabe umfasst ca. 15 Minuten, wobei oftmals unvollständige Quelltexte bereits vorgegeben werden, die von den Teilnehmern gemäß der Aufgabenstellung zu vervollständigen sind.

Auf diese Weise können auch umfangreiche Aufgaben, die beispielsweise eine bestimmte Klassenhierarchie oder eine halbfertige GUI-Applikation voraussetzen, angemessen als Übungsaufgabe behandelt werden. Die Teilnehmer arbeiten in diesem Szenario weitestgehend selbständig, werden aber durch Tutoren betreut, die sie bei Bedarf hinzuziehen können. Den Abschluss einer Übungsaufgabe bildet die Präsentation einer Musterlösung (inkl. Erläuterungen) durch den Dozenten.

7.1.2.3 Tutorien

Ergänzend zu jeder Vorlesungseinheit besteht für die Studierenden die Möglichkeit, ein Tutorium zu besuchen. Diese regelmäßig einmal pro Woche stattfindende

Übungseinheit von 90 Minuten Dauer gibt den Teilnehmern die Möglichkeit, unter Betreuung durch Tutoren (s. Kap.5.7.2) die zuvor vermittelten Lehrinhalte zu wiederholen, zu diskutieren und vor allem praktisch umzusetzen. Das Spektrum der Aufgaben reicht von den Beispielen und Übungsaufgaben, die unter Betreuung wiederholt beziehungsweise vervollständigt werden können, über eigens für ein Tutorium konzipierte Aufgaben bis hin zu den Hausaufgaben (oder Hausübungen; dazu mehr in Kap.7.1.2.4).

Die Aufgabenbearbeitung im Rahmen von Tutorien findet üblicherweise in Form von Gruppenarbeit an Rechnern statt. Diese Arbeitsform und der gegenüber den Beispielen oder den Übungsaufgaben deutlich längere Zeitrahmen erlauben es, in Tutoriumsaufgaben mehrere Lehrinhalte einer zugehörigen Vorlesungseinheit und damit assoziierte Elemente der zu erlernenden Programmiersprache zu behandeln. Somit ergänzen die Tutorien die Vorlesungseinheiten um praktische Übungseinheiten, die den Teilnehmern angemessen Gelegenheit bieten, ihre für die Programmierung unerlässlichen praktischen Fertigkeiten zu erlangen und zu festigen.

7.1.2.4 Hausaufgaben

In einer Hausaufgabe werden mehrere – üblicherweise zwei bis drei – Vorlesungseinheiten mitsamt den dort behandelten Themen zusammengefasst und weitestgehend regelmäßig im Verlauf einer Vorlesung angeboten. Die Bearbeitung durch die Teilnehmer erfolgt – wie bereits bei den Tutorien – in Form von Gruppenarbeit, im Gegensatz zu den erwähnten Tutorien entfällt allerdings die direkte und unmittelbare Betreuung durch Tutoren, da die Bearbeitung von Hausaufgaben durch eine Gruppe zunächst räumlich und zeitlich von der Vorlesung und zugehörigen Tutorien entkoppelt stattfindet.

Erfahrungsgemäß treffen sich diese Gruppen beispielsweise am Wochenende und arbeiten an ihren privaten Rechnern zusammen oder aber sie nutzen die Zugänge zu den Rechnerräumen ihrer Universität in den Zeiten, in denen die Räume frei zugänglich und nicht durch Veranstaltungen blockiert sind. Ergänzend können auch die Tutorien für die Bearbeitung von Hausaufgaben genutzt werden. Dies bietet sich insofern an, als dass in Problemsituationen einer der Tutoren um Rat gefragt werden kann.

Als weitere Praxisform dienen auch die Hausaufgaben dem Ziel, die für die Programmierung in einer Sprache wie Java notwendigen praktischen Fertigkeiten zu erlangen und zu festigen. Darüber hinaus erfolgt auf Basis der Hausaufgaben auch eine Überprüfung der Lernziele: Teilnehmer reichen die von ihnen bearbeiteten Hausaufgaben bei einem der Tutoren ein, woraufhin eine Bewertung durch den Tutor nebst Vergabe von Leistungspunkten erfolgt. Diese wiederum stellen eine Art Zulassungskriterium für die Teilnahme am Abschlussprojekt dar.

7.1.2.5 Abschlussprojekt

Den Abschluss einer Programmierveranstaltung bildet das Abschlussprojekt, anhand dessen die Leitsungsbewertung der einzelnen Teilnehmer erfolgt. Die Bearbeitung erfolgt analog zu Tutorien und Hausaufgaben in Gruppen und ist zeitlich in der vorlesungsfreien Zeit nach der letzten Vorlesungseinheit angesiedelt, so dass den Teilnehmern für die Bearbeitung ein Zeitraum von etwa acht Wochen zur Verfügung gestellt werden kann. Inhaltlich werden mit Abschlussarbeiten weite Bereiche abgedeckt, die von Lernprogrammen über Spiele bis zu Programmen zur Verwaltung von Haushaltsbudgets, Rezepten oder auch CD-Sammlungen reichen. Nach Abgabe der fertigen Arbeit erfolgt zu einem vereinbarten Termine eine Präsentation des laufenden Projekts durch die Gruppe sowie ein mündlicher Prüfungsteil, der sowohl allgemeine Fragen zu Lehrinhalten der Vorlesung als auch spezielle Fragen zu ausgewählten Stellen in den Quelltexten des Projekts umfasst. Zusammen mit einer Bewertung der Quelltexte anhand diverser Kriterien (wie Formatierung des Codes, Umfang und Inhalt von Kommentaren, sinnvolle Verwendung einzelner Sprachelemente, Design des Programms etc.) ergibt sich eine Gesamtnote für jeden Teilnehmer.

Diese Vorgehensweise ist für beide Parteien vorteilhaft:

- Der Dozent kann sicherstellen, dass möglichst viele Aspekte der zu lernenden Programmiersprache in dem Abschlussprojekt Verwendung finden, das heißt dass sich ein möglichst hoher Abdeckungsgrad hinsichtlich der in der Vorlesung behandelten Themen ergibt.

- Die Studierenden erhalten einen angemessenen zeitlichen Rahmen, der es ihnen ermöglicht, bei weitgehend freier Zeiteinteilung ein zwar umfangreiches, aber dafür auch attraktives Projekt zu erstellen.

7.1.3 Didaktische Aspekte

Das in dieser Unterrichtsform zum Einsatz kommende didaktische Konzept ist geprägt vom konstruktivistischen Gedanken, nach welchem Wissen nicht einfach im Hirn des Lerndenden abgelagert (Behaviourismus) oder dort *nur* verarbeitet (Kognitivismus) wird, sondern vom Lernenden selber in Form eines gleichsam aktiven wie sozialen Prozesses und in Wechselwirkung mit seiner Umwelt generiert wird (s. Kap.5.4.1; vgl. Baumgartner & Payr, 1994). Dementsprechend besteht das Lernziel nicht darin, richtige Antworten geben zu können (Behaviourismus) oder die richtige Methode zur Antwortfindung anwenden zu können (Kognitivismus), sondern vor allem in der Bewältigung auch komplexer Situationen und Probleme. Die Erreichung dieses Lernziels wird unterstützt mittels entsprechend formulierter Übungsaufgaben: Die Einbettung der Problemstellung in einen authentischen Kontext stellt aus konstruktivistischer Sicht eine wesentliche Forderung an eine erfolgversprechende Lernsituation dar (s. Kap.5.4.3).

Für das vorliegende Lernszenario bedeutet dies die Formulierung von Aufgaben, die mögliche Alltagsprobleme der Zielgruppe abdeckt wie beispielsweise die Mietberechnung für eine von einer Studierenden-WG genutzten Mehrzimmerwohnung, die Erstellung einer Einkaufsliste auf Basis gegebener Rezepte und der Anzahl am Essen teilnehmender Personen (mit Möglichkeiten zur Optimierung, sofern mehrere Supermärkte mit unterschiedlichen Preisen verfügbar sind) oder die Umsetzung eines einfachen Fahrkartenautomatens, aber auch einfache Spiele wie Knobeln und Galgenraten haben sich als geeignet erwiesen.

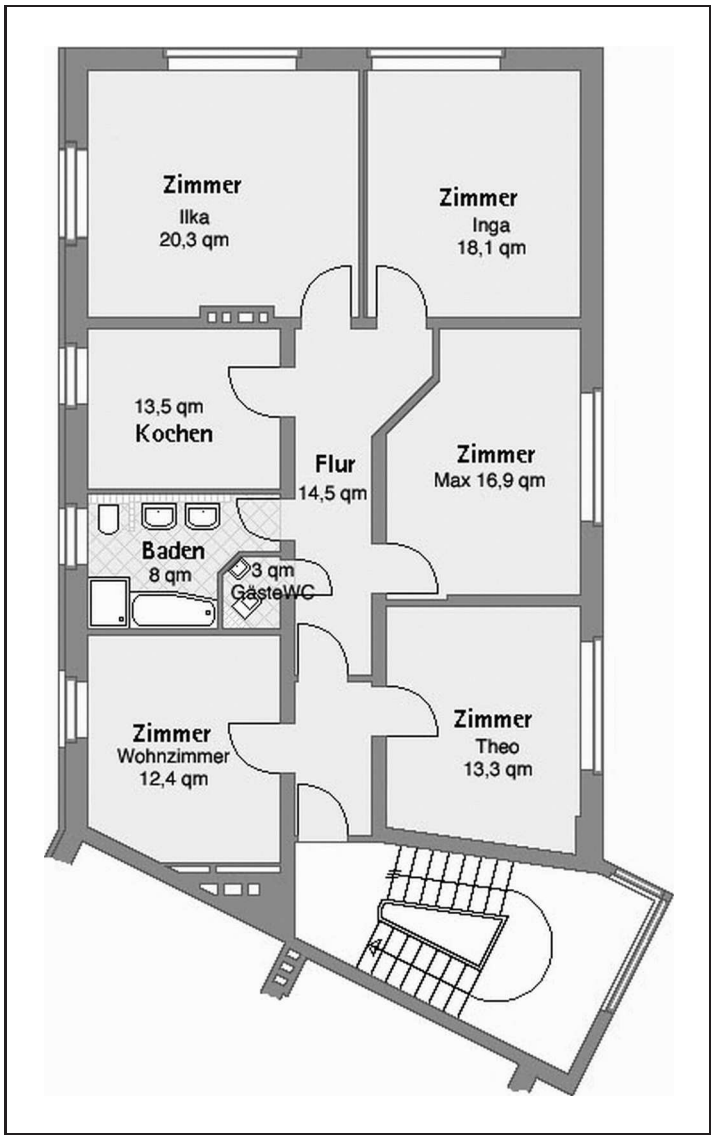

Abb. 7.1: Situiertes, problembasiertes Lernen: Mietberechnung einer WG

Prinzipiell hat sich diese Kombination aus Theorievermittlung und praktischen Übungseinheiten mit Bezug zu Alltagsproblemen bei der Durchführung von Lehrveranstaltungen zu Programmiersprachen erfolgreich gezeigt, was durch eine Vielzahl qualitativ hochwertiger Projekte dokumentiert ist. Trotzdem können regelmäßig Probleme bei der praktischen Bearbeitung von Übungen aller zuvor ge-

nannten Kategorien beobachtet werden.

7.2 Die ersten Fehler - eine Fallstudie

Die praktische Umsetzung von zuvor theoretisch behandelten Lehrinhalten bereitet vielen Studierenden bereits mit der allerersten Übungsaufgabe Probleme. Zu Beginn des Studienjahres 2001/2002 notierten die Tutoren während der Übungseinheit, die in die erste Vorlesungseinheit integriert ist und deren letzte Viertelstunde beansprucht, sämtliche Problemsituationen, zu denen sie von den Studierenden zwecks Hilfestellung hinzugezogen wurden. An zwei voneinander unabhängigen Terminen wurde dieselbe Vorlesungseinheit vor jeweils 30 Studierenden gehalten, so dass sich insgesamt 60 Teilnehmer an dieser Fallstudie beteiligten.

7.2.1 Die Aufgabe: „Hello, World!"

Der theorievermittelnde Teil dieser Vorlesungseinheit endet mit einem kleinen Java-Programm, das mittels einer einfachen Ausgabeanweisung den Text *„Hello, world!"* in der Shell ausgibt. Der gesamte Quelltext (inkl. Kommentare) sowie die notwendigen Schritte zum Erstellen, Übersetzen und Ausführen des Programms werden auf zwei aufeinanderfolgenden Folien dem Auditorium präsentiert.

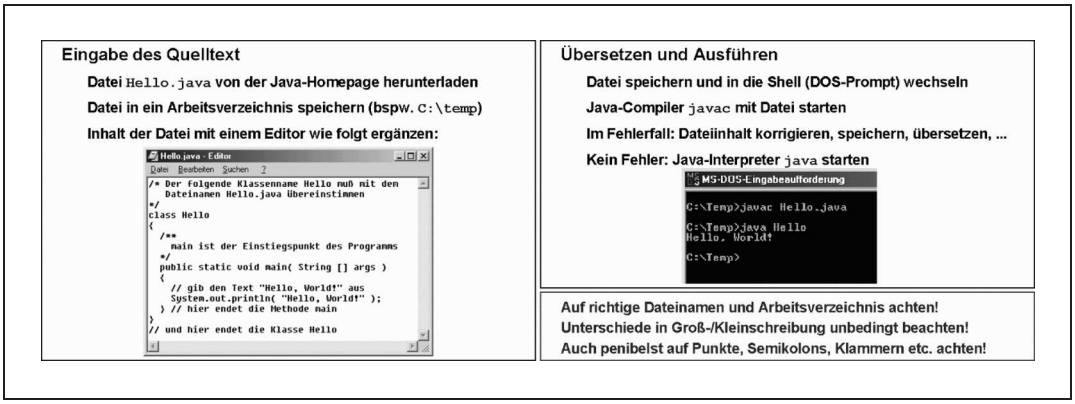

Abb. 7.2: Vorlesungsfolien der 1. Programmieraufgabe „Hello, World!"

Die Aufgabe besteht nun darin, dieses erste Java-Programm selbständig zu erstellen und zur Ausführung zu bringen. In Anbetracht des knappen zeitlichen Rahmens von maximal 15 Minuten für die Bearbeitung dieser Aufgabe und unter Berücksichtigung der für die Teilnehmer in der Regel noch ungewohnten Situation wird den Lernenden eine Textdatei zur Verfügung gestellt, die bereits Teile des zu erstellenden Quelltexts in Form von Kommentaren und Klammerstrukturen enthält[1].

[1] Die Zeilennummern in den hier aufgeführten Quelltextbeispielen dienen lediglich der besseren Orientierung und sind nicht Bestandteil der Originalquelltexte.

Beispiel 7.1: Unvollständiger Quelltext von ,,Hello,World!"

```
1 /* Der fogende Klassenname Hello muss mit dem
2    Dateinamen Hello.java uebereinstimmen
3  */
4 class
5 {
6   /**
7    main ist der Einstiegspunkt des Programms
8    */
9   main
10  {
11    //gib den Text "Hello, World!" aus
12
13  } // hier endet die Methode main
14 } // und hier endet die Klasse Hello
```

■

Dieses Textgerüst ist gemäß der nachfolgenden Vorlage (s. Beispiel 7.2) zu vervollständigen und abzuspeichern.

Beispiel 7.2: Kompletter Quelltext von ,,Hello, World!"

```
1 /* Der fogende Klassenname Hello muss mit dem
2    Dateinamen Hello.java uebereinstimmen
3  */
4 class Hello
5 {
6   /**
7    main ist der Einstiegspunkt des Programms
8    */
9   public static void main ( String args[] )
10  {
11    //gib den Text "Hello, World!" aus
12    System.out.println( "Hello, World!" );
13  } // hier endet die Methode main
14 } // und hier endet die Klasse Hello
```

■

Konkret bedeutet dies folgende Ergänzungen gegenüber der Vorlage:

- Der Klassenname `Hello` ist in Zeile 4 nach dem Schlüsselwort `class` zu notieren.
- In Zeile 9 ist der Kopf der Methode `main` zu komplettieren.
- Die gesamte Ausgabeanweisung muss in Zeile 12 korrekt wiedergegeben werden.

Im Anschluss an diese Ergänzungen wird – mittels Übersetzung durch den Java-Compiler – der zuvor gespeicherte Quelltext in den Ziel-Code überführt. Sofern bis zu diesem Schritt keine Fehler aufgetreten sind, ist eine abschließende Programmausführung mittels des Java-Interpreters möglich (s. Beispiel 7.3).

Beispiel 7.3: Übersetzung und Programmausführung von „Hello,World!"

```
>javac Hello.java

>java Hello
Hello, World!

>_
```

■

Während der Bearbeitung dieser Aufgabe standen allen Teilnehmern zwei Tutoren zur Verfügung, die in Problemsituationen um Rat gefragt werden konnten.

7.2.2 Beobachtete Anfängerfehler

Obwohl bei dieser allerersten Programmieraufgabe lediglich ein gegebener Text mit einer Länge von ungefähr 14 Zeilen zu reproduzieren war – und damit der Anspruch zumindest aus Sicht der Software-Entwicklung vernachlässigbar gering ist –, konnten 13 fehlerhafte Quelltexte, ein falscher Dateiname sowie in zwei weiteren Fällen je eine Fehlbedienung des Java-Compilers und des Java-Interpreters registriert werden.

7.2.2.1 Syntaktisch falsche Schlüsselwörter

Schlüsselwörter gehören zu den Grundsymbolen einer jeden Programmiersprache und besitzen eine feste unveränderbare Bedeutung (vgl. Goos & Zimmermann, 1999, S.481). Insbesondere müssen Schlüsselwörter bei ihrer Verwendung exakt gemäß ihrer Definition notiert werden. Dass dies – speziell am Anfang des Erlernens einer bis dato unbekannten Programmiersprache – nicht immer gelingt, konnte gleich drei mal beobachtet werden. So notierte beispielsweise einer der Teilnehmer `station` anstatt `static`:

Beispiel 7.4: `station` statt `static`

```
...
public station void main ( String args[] )
...
```

■

Der Versuch, den fehlerhaften Quelltext zu übersetzen, resultiert in folgenden Fehlermeldungen:

Beispiel 7.5: Übersetzen fehlerhafter Schlüsselwörter

```
>javac Hello.java
Hello.java:9: <identifier> expected
  public station void main ( String args[] )
                 ^
Hello.java:13: ';' expected
  }  // hier endet die Methode main
   ^
```

```
2 errors
```
■

Ein anderer Teilnehmer schrieb `statis` anstelle von `static`, ein dritter Teilnehmer verfälschte das Schlüsselwort `public` zu `publis`. In allen Fällen erhielten die Teilnehmer Fehlermeldungen wie in Beispiel 7.5.

7.2.2.2 Fehlerhaft notierte Klassennamen

Ohne die zentrale Rolle, die Klassen in objektorientierten Sprachen spielen, schmälern zu wollen, können Klassen vereinfacht als Datentypen betrachtet werden, die einerseits definiert werden und andererseits zur Deklaration und Definition von Daten (genauer: von Objekten) herangezogen werden. Bei der Definition einer Klasse wird dieser (nach dem Schlüsselwort `class`) ein Klassenname in Form eines sogenannten Bezeichners (*Identifier*) zugeordnet. Da in der Programmiersprache Java strengstens zwischen Groß- und Kleinschreibung unterschieden wird, ist es unerlässlich, Bezeichner (wie bspw. Klassennamen) exakt gemäß ihrer ursprüngliche Definition zu verwenden. Darüber hinaus ist es bei der Programmierung in Java Konvention, dass die zu einer Klasse gehörende Quelldatei mit demselben Namen beginnt und die Dateiendung `.java` besitzt.

Eine neue Klasse (namens `Hello`) wird auch im Rahmen der vorliegenden Aufgabe definiert. Zusätzlich wird – im Header der Methode `main` – die vordefinierte Klasse `String`[2] verwendet. Gemäß oben genannter Konvention wird in der Datei `Hello.java` die Definition der Klasse `Hello` erwartet; diese Konvention wird auch durch die Vorlage (s. Beispiel 7.2) erfüllt.

Trotzdem haben gleich zwei der Teilnehmer unabhängig voneinander den Klassenname komplett in Kleinbuchstaben notiert:

Beispiel 7.6: Falscher Name bei Definition der Klasse `Hello`

```
...
class hello
...
```
■

Da in diesem Beispiel lediglich eine Vereinbarung verletzt wird, ist der Quelltext als solcher weiterhin syntaktisch korrekt und lässt sich anstandslos und ohne Fehlermeldung vom Compiler übersetzen. Den Versuch, das Programm dem Java-Interpreter zur Ausführung zu übergeben, beantwortet dieser mit Fehlermeldungen wie in nachfolgendem Beispiel:

Beispiel 7.7: Ausführung einer Klasse mit falschem Namen

```
>javac Hello.java

>java Hello
Exception in thread "main" java.lang.NoClassDefFoundError: Hello
```

[2] Die Klasse `String` beziehungsweise `java.lang.String` ist fester Bestandteil jeder Java-Umgebung.

```
        (wrong name: hello)
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClass(Unknown Source)
        at java.security.SecureClassLoader.defineClass(Unknown Source)
        at java.net.URLClassLoader.defineClass(Unknown Source)
        at java.net.URLClassLoader.access$100(Unknown Source)
        at java.net.URLClassLoader$1.run(Unknown Source)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at java.lang.ClassLoader.loadClassInternal(Unknown Source)

>_                                                                    ■
```

Zwar enthalten die Fehlermeldung des Interpreters gleich in den ersten beiden Zeilen klare Information über die Fehlerursache, jedoch konnte beobachtet werden, dass die nachfolgenden Ausgabezeilen viele Programmieranfänger irritieren, da sie schlichtweg zuviele Informationen enthalten, deren Bedeutung sich einem Anfänger ohne weiteres Zutun (zum Beispiel durch einen Tutor, der Erklärungen liefert) nicht erschließt. In der Folge zeigten sich viele Teilnehmer bei Auftreten dieses Fehlers einfach überfordert.

Ein gänzlich anderes Fehlverhalten ergibt sich, wenn eine Klasse bei ihrer Verwendung falsch geschrieben wird. Dies demonstriert folgendes Beispiel 7.8, bei welchem ein Teilnehmer die Klasse `String` fälschlicherweise als `string` notiert:

Beispiel 7.8: Falscher Name bei Verwendung einer Klasse

```
...
public static void main ( string args[] )
...                                                                  ■
```

In diesem Fall erzeugt bereits der Compiler eine Fehlermeldung und weist den Anwender relativ genau auf die Fehlerquelle und den Fehlergrund hin:

Beispiel 7.9: Übersetzung bei Verwendung eines falschen Klassennamens

```
>javac Hello.java
Hello.java:9: cannot find symbol
symbol  : class string
location: class Hello
  public static void main ( string args[] )
                            ^
1 error

>_                                                                   ■
```

Wie an diesen beiden Beispielen aufgezeigt werden konnte, resultiert die falsche Notation eines Klassennamens bei Definition und Verwendung in gänzlich unterschiedlichen Fehlersituationen.

7.2.2.3 Fehlerhafte notierte Methodennamen

Methoden als Operationen einer Klasse werden innerhalb ihrer Klasse definiert. Dabei wird – ähnlich wie bei den Klassennamen – jeder Methode ein Bezeichner zugeordnet, der beim Aufruf der Methode Verwendung findet. Auch hierbei ist strengstens auf die richtige Schreibweise zu achten, da sonst ähnliche Fehler wie bei der Verwendung von falschen Klassennamen (s. Kap.7.2.2.2) beobachtet werden können.

In der vorliegenden Fallstudie verfälschten zwei Teilnehmer den Methodennamen `println` der Ausgabeanweisung `System.out.println(...)` (s. Beispiel 7.2, Zeile 12) und notierten stattdessen `printeln` beziehungsweise `printIn`, was in beiden Fällen vom Compiler mit nahezu identischer Fehlermeldung quittiert wurde.

Beispiel 7.10: Übersetzung bei Verwendung eines falschen Methodennamens

```
>javac Hello.java
Hello.java:12: cannot find symbol
symbol  : method printeln(java.lang.String)
location: class java.io.PrintStream
    System.out.printeln( "Hello, World!" );
               ^
1 error

>_
```

In einem Fall *korrigierte* einer der Teilnehmer den bereits vorgegebenen Methodennamen `main` zu `man`, was zur Folge hat, dass das auszuführende Programm keine `main`-Methode[3] besitzt. Dies führt allerdings erst bei der Ausführung durch den Interpreter zu einem Fehler:

Beispiel 7.11: Ausführung einer Klasse ohne `main`-Methode

```
>javac Hello.java

>java Hello
Exception in thread "main" java.lang.NoSuchMethodError: main

>_
```

Auch diese Beispiele zeigen deutlich auf, dass die Einhaltung von Syntax und Vereinbarungen bei der Programmierung unerlässlich sind.

7.2.2.4 Fehlender Rückgabetyp einer Methodendefinition

Im Zusammenhang mit der `main`-Methode konnte bei einem der Teilnehmer ein weiterer Fehler beobachtet werden, der durch Weglassen des Schlüsselwortes `void` entstand:

[3] Jedes Java-Programm muss eine Methode namens `main` mit passender Signatur (s. Beispiel 7.2, Zeile 9) besitzen, da diese vom Interpreter als Einstiegspunkt in die Programmausführung erwartet wird.

Beispiel 7.12: Übersetzung einer Methode ohne Rückgabetyp

```
>javac Hello.java
Hello.java:9: invalid method declaration; return type required
  public static main ( String args[] )
                ^
1 error

>_
```

■

7.2.2.5 Fehlendes Punktzeichen

Der Punkt . wird in der Programmierpsrache Java als sogenannter *Zugriffsoperator* interpretiert, der den Zugriff auf Elemente (in Form von Attributen und Methoden) von Klassen und Objekten regelt. Dieser Operator kommt in der vorliegenden Aufgabe gleich zweimal innerhalb einer Anweisung zur Anwendung (s. Beispiel 7.2, Zeile 12), wurde aber von einem der Teilnehmer in einem Fall durch ein Leerzeichen ersetzt.

Beispiel 7.13: Fehlendes Punktzeichen

```
...
System.out println( "Hello, World!" );
...
```

■

Daraus resultiert folgender Übersetzungsfehler:

Beispiel 7.14: Übersetzungsfehler bei fehlendem Punktzeichen

```
>javac Hello.java
Hello.java:12: ';' expected
    System.out println( "Hello, World!" );
                       ^
1 error

>_
```

■

Interessanterweise wird dem (unerfahrenen!) Anwender durch diese Fehlermeldung suggeriert, dass er ein Semikolon ; vergessen hat.

7.2.2.6 Fehlendes Zeilenende

Zeilenende stellen – wie beispielsweise auch Leerzeichen und Tabulatoren – zunächst Hilfsmittel dar, die zur Formatierung von Quelltexten und damit verbunden zu einer übersichtlicheren Darstellung des Quellcodes verwendet werden. Aus Sicht des Compilers sind diese Zeichen in vielen Fällen überflüssig und werden daher bei der Übersetzung in den Byte-Code durch Überlesen ignoriert.

Eine Ausnahme bilden die sogenannten Zeilenkommentare: Wenn der Compiler innerhalb eines Quelltexts auf die Zeichenfolge // trifft, interpretiert er sämtliche bis zum nächsten Zeilenende auftretenden Zeichen als Kommentar und überliest

diese. Im Rahmen der vorliegenden Fallstudie konnte bei zwei Teilnehmern beobachtet werden, dass sie die Ausgabeanweisung an den ersten Zeilenkommentar (s. Beispiel 7.2, Zeile 11) anhängten, anstatt diese in einer neuen Zeile zu notieren. Das Ergebnis ist im folgenden Beispiel zu sehen.

Beispiel 7.15: Fehlendes Zeilenende

```
...
// gib den Text "Hello, World!" aus System.out.println( "Hello, World!" );
...
```
■

Aufgrund des fehlenden Zeilenendes wird auch erwähnte Ausgabeanweisung als Kommentar interpretiert und vom Compiler verworfen. Syntaktisch ist der Quelltext damit absolut korrekt, das heißt er lässt sich fehlerfrei übersetzen und ausführen.

Beispiel 7.16: Ausführung mit fehlendem Zeilenende

```
>javac Hello.java

>java Hello

>_
```
■

Trotz dieser an sich fehlerfreien Ausführung muss das Programm dennoch als problembehaftet angesehen werden, da das Verhalten bei der Ausführung stark vom erwarteten Programmverhalten abweicht: Anstelle der Textausgabe von *„Hello, World!"* erzeugt das Programm gar keine Ausgabe, aber eben auch keine Fehlerausgabe.

Zusätzlich zu den bislang genannten Fehlern konnten zwei weitere Probleme beobachtet werden, die nicht auf falsche Schreibweise oder fehlerhafte Verwendung der Sprachelemente von Java zurückgeführt werden können, sondern bei der Bedienung der verwendeten Software-Werkzeuge (Editor, Compiler und Interpreter) zutage treten.

7.2.2.7 Bedienfehler

Die Bearbeitung von Quelltexten erfordert einen Texteditor, der in der Lage ist, ASCII-Texte (bzw. ANSI-Texte) zu editieren. Unter einem gängigen Windows-Betriebssystem der Firma Microsoft kann dies im einfachsten Fall vom Programm `notepad` geleistet werden, das zu einer Standardinstallation des Betriebssystems gehört. Dieser Editor verarbeitet üblicherweise Textdateien, also solche Dateien, die die Dateiendung `.txt` besitzen. Bei unachtsamer Bedienung dieses Editors kann es vorkommen, dass beim Abspeichern einer beliebigen ASCII-Datei (wie eben beispielsweise eines Java-Quelltexts) eben diese Dateiendung an den eigentlichen Dateinamen angehängt wird (s. Abb.7.3).

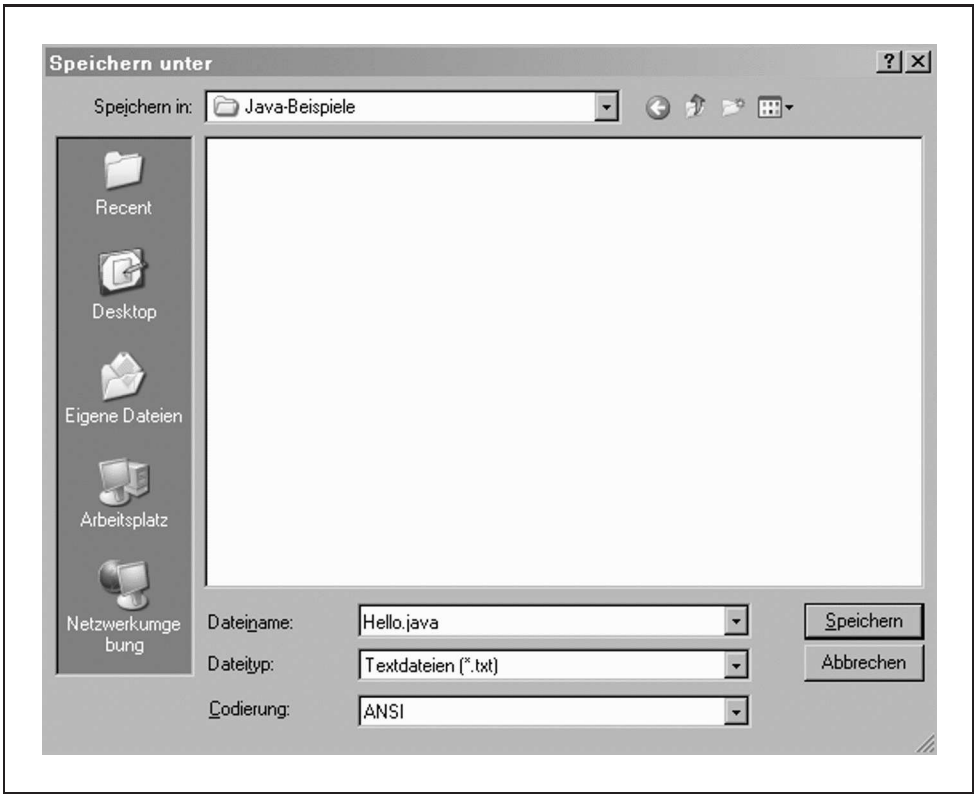

Abb. 7.3: Automatische Dateiendung `.txt` beim Speichern eines Quelltexts (Screenshot)

Dies ist auch einem der Teilnehmer passiert, der daraufhin in seinem Arbeitsverzeichnis statt der erwarteten Datei `Hello.java` die Datei `Hello.java.txt` vorfand. Der Versuch, den Compiler zu starten, führte zu folgendem Fehler:

Beispiel 7.17: Compiler-Fehler bei nicht vorhandener Quelltext-Datei

```
>javac Hello.java
error: cannot read: Hello.java
1 error
>_
```

■

Zur selben Fehlermeldung führte ein Tippfehler beim Aufruf des Compileres, als einer der Teilnehmer in der Shell den Befehl `javac Hell.java` anstelle von `javac Hello.java` ausführen lassen wollte.

Beim Start des Java-Interpreters ist üblicherweise der Name der auszuführenden Klasse anzugeben. Im vorliegenden Fall würde der Befehl `java Hello` dies leisten. Einer der Teilnehmer ergänzte dies jedoch um die Dateiendung `.java`. Der resultierende Befehl lautete daraufhin `java Hello.java`:

Beispiel 7.18: Fehlerhafter Aufruf des Java-Interpreters

```
>java Hello.java
Exception in thread "main" java.lang.NoClassDefFoundError: Hello/java
>_
```

■

7.3 Unterstützung in Problemsituationen

Zunächst ist festzuhalten, dass bei den 60 Teilnehmern in 16 Fällen Schwierigkeiten beobachtet werden konnten bei der Bearbeitung der Aufgabe, einen gegebenen Quelltext fehlerfrei zu reproduzieren und zur Ausführung zu bringen. Es ist offensichtlich, dass sich die Teilnehmer mit einer für sie ungewohnten Situation konfrontiert sehen, die eine Vielzahl potentieller Probleme beinhalt.

7.3.1 Problemursachen

Die Ursachen für die Probleme beim Programmieren sind nahezu so vielfältig wie die Studien, die sich mit diesem Thema befassen. Bischoff (2005) verweist auf eine Vielzahl relevanter Arbeiten und legt deren Ergebnisse ausführlich dar (vgl. Bischoff, 2005, Kap.2.3). Nachfolgend werden die für die vorliegende Arbeit wichtigsten Erkenntnisse zusammengefasst.

7.3.1.1 Programmieren als Problem

Die prinzipiellen Ursachen für Problemsituationen während des Programmierens finden sich in Bischoff (2005) treffend zusammengefasst: *„Das Erlernen einer Programmiersprache stellt für Anfänger oft eine große Hürde dar.* du Boulay *(1989) nennt als Ursache dafür fünf Problemfelder, die es gleichzeitig zu bewältigen gilt: (1) Generelle Orientierung, wozu man Programme erstellt und welche Vorteile man aus ihnen ziehen kann, (2)* „the notional machine"*, das Entwickeln eines adäquaten Models des Computers bezüglich der Vorgänge bei der Verarbeitung von Programmen, (3) Notation, die Syntax und Semantik einer konkreten Sprache, (4) Strukturen, das heißt Schemata im Sinne wiederkehrender Codemuster zur Erreichung bestimmter Pläne und (5) die Pragmatik des Programmierens, die Fähigkeit Programme ggf. mithilfe verschiedener Tools zu planen und zu entwickeln, zu testen und zu debuggen"* (Bischoff, 2005, S.12).

Insbesondere lassen sich durch das (noch) fehlende Wissen über *„programmiersprachliche Notationen und Konzepte"* (vgl. Bischoff, 2005, Kap.2.3.1) viele der oben genannten Probleme erklären, die schlichtweg auf fehlerhafte Schreibweisen von Schlüsselwörtern und Bezeichnern zurückzuführen sind. Dies darf nicht weiter verwundern, da den Anfängern einer (Programmier-)Sprache der Bezug zu den in dieser Sprache verfassten (Quell-)Texten und den dort verwendeten Sprachelementen größtenteils fehlt. Entsprechend problembehaftet ist auch die Reproduktion eines Textes, der ja in einer zunächst unbekannten Sprache vorliegt. Die resultierenden Fehlermeldungen von Compiler und Interpreter tragen nur in wenigen Fällen zur Klärung der Problemsituation bei. So stellt auch beispielsweise der Hinweis des Compilers auf ein erwartetes – und daher scheinbar fehlendes – Semikolon in Folge eines falsch notierten Schlüsselwortes (s. Beispiele 7.4 und 7.5) keine brauchbare Hilfe dar. Entsprechend häufig kann beobachtet werden, dass Tutorien bei auftretenden Compiler-Meldungen hinzugezogen werden, um diese verständlich

zu machen, das heißt für die Teilnehmer geeignet zu übersetzen und zu erläutern, so dass der Fehler anschließend von den Teilnehmern relativ problemlos lokalisiert und behoben werden konnte. Diese reine Übersetzung oder auch Aufbereitung von Fehlermeldungen des Compilers stellt somit ein nicht zu vernachlässigende Hilfe gerade für Anfänger dar.

7.3.1.2 Der Rechner als Problemquelle

Speziell die in Kapitel 7.2.2.7 beobachteten Probleme sind gekennzeichnet durch Unsicherheiten und mangelnde Praxis bei der korrekten, zielgerichteten Bedienung des Arbeitsplatzrechners, da das Arbeiten mit einer Shell in Zeiten graphisch-basierter Benutzungsoberflächen kaum noch in dieser Form praktiziert wird. Hier allerdings hat die Durchführung eines Vorbereitungskurses in nachfolgenden Semestern positive Effekte gezeigt: In der Einführungswoche werden vor der ersten Vorlesungseinheit in einem in etwa dreistündigen Intensivkurs grundlegende Konzepte der Rechnerbedienung (am Beispiel einer *Windows 2000*-Umgebung) per Shell vermittelt und anhand kleiner Aufgaben geübt. Dazu gehören das Starten einer Shell, das Navigieren innerhalb von Verzeichnisstrukturen sowie das Erzeugen, Kopieren, Umbenennen und Löschen von Verzeichnissen und Dateien mit entsprechenden Shell-Befehlen. Die Teilnehmer dieses Kurses zeigen sich in den nachfolgenden Vorlesungseinheiten nicht nur sicherer im Umgang mit dem Rechner, sondern machen auch geringeren Gebrauch vom Unterstützungsangebot der Tutoren bei der allerersten Übungsaufgabe *„Hello, World!"*.

7.3.1.3 Das Team als Problemursache

Zusätzlich zu den eher fachlichen und technischen Problemursachen kann auch die Gruppe selber – respektive ihre Mitglieder – die Ursache für Probleme darstellen, die während der Zusammenarbeit auftreten. Dieses lässt sich weitestgehend zurückführen auf Strukturen von Kleingruppen wie soziometrische, Kommunikations- und Machtstrukturen (s. Kap.2.3), die gleichsam Chancen wie Risiken für das Miteinander in der Gruppe beinhalten. Desweiteren stellt auch die Kommunikation zwischen den Mitgliedern einer Gruppe einen potentiellen Auslöser für Probleme dar, die auf verschiedenen Kommunikationsebenen anzusiedeln sind. Göldner (2005) hat sich in ihrer Arbeit ausführlich mit diesen Problemen im Kontext der objektorierentierten Programmierung auseinandergesetzt.

7.3.2 Hilfestellungen

Trotz der genannten Vorbereitungskurse bleibt das Erlernen einer Programmiersprache wie Java allein schon aufgrund der in Kapitel 7.3.1.1 skizzierten Gründe

problembehaftet. Diese Unvermeidbarkeit von Problemsituationen bedingt ein umfangreiches Angebot an verschiedenartigen Hilfestellungen, auf die die Lernenden bei Bedarf zugreifen können.

7.3.2.1 Informationsquellen bei der Programmierung

Die primäre Hilfestellung im Rahmen einer Vorlesung zur Programmierung erhalten die Lernenden in Form eines Skriptums, welches neben den zu vermittelnden Lehrinhalten auch entsprechende Beispiele enthält und durch passende Übungsaufgaben (mitsamt lauffähigen Lösungsvorschlägen) ergänzt wird. Im Rahmen der Tutorien erhalten die Teilnehmer bei Bedarf weitere Hilfe durch die Tutoren (s. Kap.7.1.2.3). Um nun aber in Erfahrung zu bringen, welche Informationsquellen von den Lernenden in Problemsituationen tatsächlich genutzt werden, wurde eine kleine Umfrage unter den Teilnehmern einer Programmierveranstaltung durchgeführt. Auf die Frage *„Wie helfen Sie sich normalerweise weiter, wenn Sie bei einer Java-Aufgabe Probleme haben?"* antworteten 27 Teilnehmer und benannten die von ihnen bevorzugten Informationsquellen. Da Mehrfachnennungen möglich waren, wurden die einzelnen Punkte in der Reihenfolge ihrer Nennung bewertet (1 = erste Nennung, 2 = zweite Nennung etc.). Das Ergebnis ist in der folgenden Tabelle zusammengefasst.

Tab. 7.1: In Problemsituationen genutzte Informationsquellen

Teilnehmer	**Buch**	**Internet**	**Kommilitone**	**Skript**	**API**	**Forum**	**Tutor**	**Skizze**
T_1	1	2	3					
T_2				1	2			
T_3	1	2						
T_4	3	2		1				
T_5	4	1		2	3			
T_6		2	3		1			
T_7				2	1			
T_8	1		2			3		
T_9	2	3		1	4			
T_{10}	1		2					
T_{11}		1						
T_{12}	1	2	3				4	
T_{13}	2			1				
T_{14}		2		1		3		
T_{15}	2		1	3				

Fortsetzung auf nächster Seite

Tab. 7.1: In Problemsituationen genutzte Informationsquellen *Forts.*

Teilnehmer	**Buch**	**Internet**	**Kommilitone**	**Skript**	**API**	**Forum**	**Tutor**	**Skizze**
T_{16}			3	1	2			
T_{17}	2	1	3					
T_{18}		2	1		3			
T_{19}	2			1				
T_{20}	2	1	3					
T_{21}				1				2
T_{22}		1						
T_{23}	2	3	1		4			
T_{24}		4	2	1	3			
T_{25}	2		3		1			
T_{26}	2		3		1			
T_{27}	1	2						
Gesamt	17	16	14	12	11	2	1	1

Die am häufigsten genannte Informationsquelle stellen demnach Bücher mit 17 Nennungen dar, gefolgt von Beispielen, die im Internet gefunden wurden (16 Nennungen), und Kommilitonen, die um Rat ersucht wurden (14 Nennungen). Erst an vierter Stelle erscheint mit zwölf Nennungen das Skript (inkl. Beispielen), dicht gefolgt von der API (elf Nennungen). Mit deutlichem Abstand dazu werden noch der Besuch eines entsprechenden Online-Forums (zwei Nennungen) sowie die Tutoren und selbsterstellte Ablaufskizzen (je eine Nennung) aufgezählt.

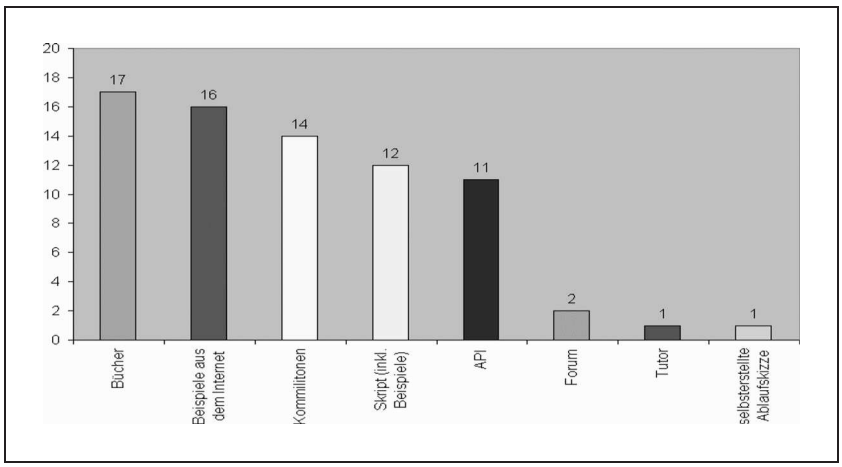

Abb. 7.4: Verwendete Informationsquellen (absolute Nennungen)

Interpretiert man zusätzlich die Reihenfolge, in der die einzelnen Punkte genannt werden, als Gewichtung nach Relevanz (wobei 1 = sehr relevant, 2 = relevant etc.)

und berücksichtigt die Gesamtzahl der Nennungen, so ergibt sich ein etwas anderes Bild (s. Abb.7.5).

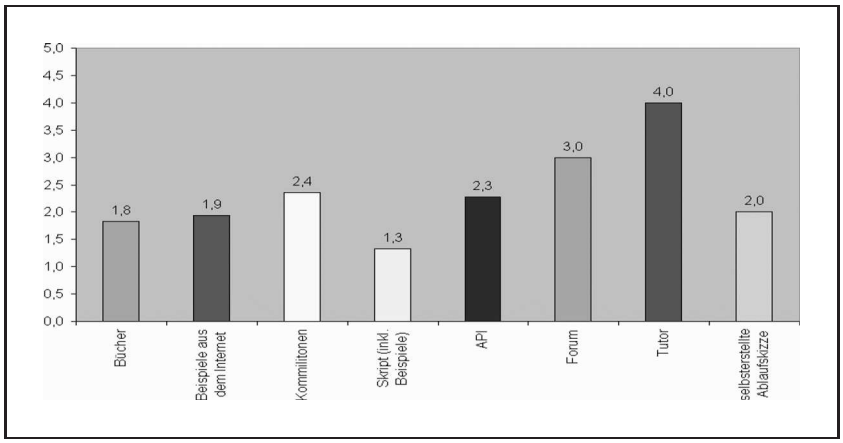

Abb. 7.5: Verwendete Informationsquellen (gewichtete Nennungen)

Mit einer durchschnittlichen Relevanz von 1,3 stellt dann das Skript (inkl. der Beispiele) die wichtigste Informationsquelle dar, gefolgt von Büchern (1,8), Beispielen aus dem Internet (1,9) und selbsterstellten Ablaufskizzen (2,0). Mit etwas Abstand werden die Recherche in der API (2,3) und die Befragung von Kommilitonen (2,4) genannt. Den vorletzten Platz bildet mit einer durchschnittlichen Nennung von 3,0 das Forum, am Schluss steht das Hilfegesuch bei einem Tutor (4,0).

Die verhältnismäßig schlechte Positionierung des Tutors mag zunächst verwundern, deckt sich aber mit dem Ergebnis einer anderen Fragestellung. Die Frage *„Welche Informationsquelle ziehen Sie bevorzugt bei Problemen zu Rate?"* besitzt die möglichen Antworten *„Menschlicher Tutor"* und *„Literatur, Internet, Skripte"* sowie vier sich dazwischen befindliche Abstufungen. Es ergibt sich eine in insgesamt sechs Stufen unterteilte Ordinalskala, auf welcher die Antwort *„Menschlicher Tutor"* den Wert 1 besitzt und der Antwort *„Literatur, Internet, Skripte"* der Wert 6 zugeordnet wird.

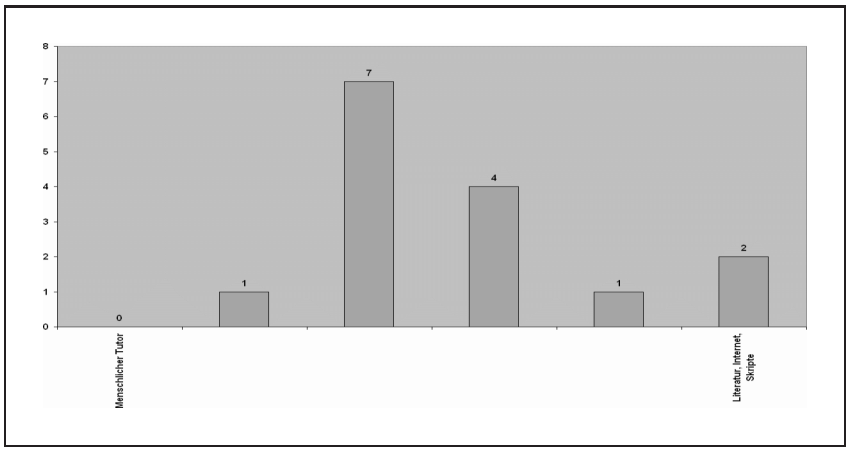

Abb. 7.6: Bevorzugte Informationsquellen

Die 15 Teilnehmer, die diese Frage beantworteten, votieren demzufolge mit einem durchschnittlichen Wert von 3,7 klar zugungsten der nicht-menschlichen Informationsquellen wie Büchern, Skripten und dem Internet. Diese Präferenz lässt sich durch zwei Umstände erklären:

1. Zunächst einmal existiert eine Hemmschwelle gegenüber einer Autorität, wie sie durch den Tutor als Experten auf dem Gebiet der Java-Programmierung dargestellt wird. Dieser Effekt kann noch verstärkt werden durch die Anwesenheit von Kommilitonen und die damit verbundene Angst, vor Publikum etwas falsches zu sagen oder eine unangebrachte Frage zu stellen (vgl. MANN, 2001, S.113ff). Hingegen tendiert die Wahrscheinlichkeit, sich einem Buch gegenüber zu blamieren, traditionell gegen Null.

2. Darüber hinaus steht der prinzipiell hohen Verfügbarkeit von Online-Medien und klassischen Print-Medien die nur begrenzte Verfügbarkeit des menschlichen Tutors gegenüber.

7.3.2.2 Das Team als Unterstützungsfunktion

Der deutlichen Bevorzugung von nicht-menschlichen Informationsquellen steht ein anderes Verhalten gegenüber, das häufig während der Bearbeitung von Übungsaufgaben beobachet werden kann: Obwohl das in Kapitel 7.1.2.2 skizzierte Szenario auf Einzelarbeit und damit auf die möglichst selbständige Bearbeitung der gestellten Aufgabe abzielt, schließen sich die Teilnehmer oftmals zu Zweiergruppen – in seltenen Fällen auch zu Dreiergruppen – zusammen, um gemeinsam eine Übungsaufgabe zu bearbeiten. In dieser besonderen Lernsituation sitzen die Teilnehmer gemeinsam vor einem einzigen Rechner und teilen sich den Zugriff auf dessen Tastatur und Bildschirm, um zusammen zu programmieren – obwohl für jeden Teilnehmer ein dedizierter Rechner zur Verfügung steht. Der in diesem Szenario nicht für die Programmierung genutzte Rechner wird stattdessen als Informationsquelle genutzt, so dass paralell zur Programmierung stets auf zugehörige Skripte, Beispiele, Online-Bücher und die API zugegriffen werden kann. Zu erwähnen sei an dieser Stelle, dass diese Form der Zusammenarbeit auch in Tutorien sowie bei der Bearbeitung von Hausaufgaben beobachtet werden kann.

Zusammenfassend ergibt sich die folgende Lernsituation: Die Mitglieder einer Gruppe bearbeiten gemeinsam eine Aufgabe und nutzen die ihnen zur Verfügung stehenden Rechner für die Programmerstellung sowie als Informationsquelle. In diesem Fall sind nicht nur die diversen Medien, sondern auch die (gleichgestellten) Teammitglieder als Informationsquelle also durchaus erwünscht.

7.3.3 Folgerungen

Die Nutzung der verfügbaren Ressourcen, insbesondere der Rechensysteme, ist unter Aspekten des CSCL (s. Kap.5) noch verbesserungsfähig:

1. Erstens ist eine starke Ortsgebundenheit festzustellen, denn alle eingesetzten Rechner müssen sich bei der soeben beschriebenen Art der Nutzung idealerweise am selben Arbeitsplatz befinden.

2. Zweitens werden die Rechner getrennt voneinander genutzt, obwohl sie untereinander vernetzt sind.

3. Drittens besitzen die Rechner statt einer aktiven lediglich eine passive Unterstützungsfunktion: Tritt eine Problemsituation auf, so muss der Prozess der Informationslieferung zunächst durch die Lernenden initiiert werden, der Rechner ist lediglich ein passiver Container, der benötigte Informationen auf Abruf bereitstellt. Auch die Auswahl der geeigneten Information erfolgt allein durch die Lernenden. Die gesamte Problemlösungsprozess wird folglich – sofern kein Tutor hinzugezogen wird – ausschließlich innerhalb der Gruppe durch deren Mitglieder durchgeführt.

Die Unterstützung von Teams bei der praktischen Umsetzung von Lerninhalten wird aus den bereits genannten Gründen als zwingend notwendig erachtet. Berücksichtigt man den für die Java-Programmierung obligatorischen Einsatz von Rechnern, so liegt es nahe, diese auch in den Lernprozess zu integrieren. Aus dem passiven Informationscontainer wird ein aktiver Informationslieferant, der Teams Unterstützung bei der Java-Programmierung und dort auftretenden Problemen anbietet. Diese Unterstützung sollte aber gezielt erfolgen, um negative Auswirkungen auf das Erreichen der Lernziele zu vermeiden. So kann es passieren, dass ein Problem aufgrund unzureichender oder ungeeigneter Unterstützung nicht von der Gruppe gelöst werden kann. Dieser Misserfolg kann zu einer Verminderung der Motivation führen (vgl. Rosemann & Bielski, 2001, S.105ff). Andererseits muss sich eine Unterstützungsfunktion auch zurücknehmen, da bei einer Überdosierung des Hilfeangebots die Gefahr besteht, dass der erwünschte Lernerfolg gänzlich ausbleibt.

Insgesamt wird also eine (a) an die Lernsituation und (b) an die Lernenden angepasste Unterstützungsfunktion in Form eines CSCL-Systems gefordert. Aus (a) folgt die Unterstützung synchroner Zusammenarbeit bei der Programmierung, so dass der grundlegende Charakter der in Kapitel 7.3.2.2 skizzierten Lernsituation erhalten bleibt: Teilnehmer einer Lerngruppe bearbeiten zeitgleich und zusammen eine gemeinsame Programmieraufgabe. Findet diese Zusammenarbeit unter ausschließlicher Nutzung einer auf vernetzten Rechnern basierenden Lernumgebung statt, so wird aus der Lerngruppe ein *virtuelles Team* (s. Kap.3.2.2). Um der in (b) genannten Forderung nach einem adaptiven CSCL-System gerecht zu werden, wird die Zusammensetzung einer Lerngruppe auf Basis eines geeigneten Rollenmodells berücksichtigt.

7.3.4 Ableitbare Forschungsfragen

Aus dieser Situation heraus ergibt sich eine Vielzahl von Fragen hinsichtlich der Unterstützung virtueller Lerngruppen bei der Programmierung in Java mittels eines

geeigneten CSCL-Systems. Der Schwerpunkt dieser Arbeit konzentriert sich auf solche Fragestellungen, die die Zusammensetzung einer Lerngruppe sowie während der Programmierung auftretende Probleme zum Inhalt haben.

7.3.4.1 Identifikation von Problemsituationen

Wie bereits in obiger Fallstudie (s. Kap.7.2) gezeigt werden konnte, ist das Erlernen einer Programmiersprache mit Problemen behaftet. Dies zeigt sich schon bei der Umsetzung einfachster Beispiele, erfährt jedoch spätestens bei der Bearbeitung von Hausaufgaben eine Steigerung. Dieses höhere Problempotential ist primär auf die gesteigerten Anforderungen – und damit verbunden die größere Komplexität – von Hausaufgaben zurückzuführen. Erschwerend kommt in dieser Lernsituation das Fehlen des Tutors hinzu: Verglichen mit Beispielen, Übungsaufgaben und Tutorien ist bei der Bearbeitung von Hausaufgaben der unmittelbare und direkte Zugriff auf den Tutor und sein Fachwissen weitestgehend auszuschließen. Entsprechendes gilt auch für die Bearbeitung des Abschlussprojekts.

Bei der Verwendung eines CSCL-Systems ist es naheliegend, den fehlenden menschlichen Tutor gegen eine entsprechende Software-Komponente, einen *virtuellen Tutor* auszutauschen. Solch ein virtueller Tutor ist integraler Bestandteil des für die Zusammenarbeit verwendeten Systems und lässt sich wie folgt skizzieren:

1. Das Team und seine Mitglieder werden bei der Zusammenarbeit durch den Tutor beobachtet, indem sämtliche Aktionen erfasst und ausgewertet werden.

2. Auf Basis der protokollierten Aktionen werden Kennzahlen als Indikatoren für Problemsituationen berechnet.

3. Kann eine Problemsituation erkannt werden, generiert der Tutor an diese Situation angepasste Unterstützung.

Voraussetzung für eine zuverlässige Entscheidung über Zeitpunkt und genaueren Typ eines Problems ist ein vollständiges Modell derjenigen Probleme, die während der gemeinsamen, synchronen Bearbeitung von Aufgaben aus der Domäne der Java-Programmierung auftreten können. Eine Sonderstellung nehmen in einem solchen Modell diejenigen Probleme ein, die zwar in dem geschilderten Kontext relevant sind, sich aber nicht oder nur sehr unpräzise durch das CSCL-System bestimmen lassen. Ein Ansatz, diesen Problemen zu begegnen, besteht darin, den Teilnehmern selbst die Möglichkeit zu geben, von sich aus Hilfe anzufordern.

Zusammenfassend sind folgende Fragestellungen zur Idenitifikation von Problemsituationen zu klären:

1. Welche Probleme treten speziell bei Anfängern der Programmierung in Java während der synchronen Zusammenarbeit unter Verwendung eines entsprechenden CSCL-Systems auf?

2. Welche dieser Probleme lassen sich während der Zusammenarbeit durch das verwendete CSCL-System erkennen?

3. Wie lassen sich erkennbare Probleme während der Zusammenarbeit identifizieren?

7.3.4.2 Teamfunktionen und Teamzusammensetzung

Im Hinblick auf eine tutorielle Komponente, die in Problemsituationen *angemessene* Unterstützung leisten soll, ist es sinnvoll, neben der Problemsituation selbst als Ursache für ein Eingreifen der tutoriellen Komponente auch die Teammitglieder als Adressaten der Unterstützung zu berücksichtigen. Die Grundlage einer differenzierten Betrachtung einer Teamzusammentsetzung bildet ein geeignetes Rollenmodell, welches – angepasst an den Kontext der objektorientierten Programmierung in Java – diejenigen Teamfunktionen definiert, die für diese spezielle Lernsituation relevant sind. Die Identifizierung der Rollen (im Sinne von Teamfunktionen) erfolgt während der Zusammenarbeit auf der Basis der Aktionen der einzelnen Teammitglieder.

Zusammenfassend sind folgende Fragestellungen hinsichtlich der Zusammensetzung virtueller Teams zu klären:

1. Welche Teamfunktionen lassen sich im Kontext der Java-Programmierung spezifizieren?

2. Welche Teamfunktionen lassen sich während der Zusammenarbeit durch das verwendete CSCL-System bestimmen?

3. Wie lassen sich diese Teamfunktionen während der Zusammenarbeit bestimmen?

In diesem Zusammenhang lässt sich unter anderem die These aufstellen: *„Das Ausmaß, in dem ein Teammitglied bestimmte Teamfunktionen wahrnimmt, lässt sich während der Zusammenarbeit anhand der von ihm ausgelösten Aktionen bestimmen"*.

7.3.4.3 Ergänzende Fragestellungen

Der erwähnte Kontext beinhaltet weitere relevante Fragestellungen, denen sich im Rahmen des Forschungsprojekts *VitaminL* andere Arbeiten widmen. Kölle (2007) befasst sich in seiner Arbeit näher mit Fragen nach Art und Inhalt tutorieller Unterstützung und erprobt unter anderem Supportstrategien unter Verwendung von CBR (*Case Based Reasoning*). Ergänzend zu den eigenen Betrachtungen von Problemsituationen werden die Ergebnisse von Bischoff (2005) und Göldner (2005) aufgegriffen, die in ihren Arbeiten spezielle Probleme während der objektorientierten Programmierung in virtuellen Teams untersucht haben, differenziert nach technischen Problemen der Java-Programmierung s. Bischoff, 2005 beziehungsweise

nach solchen Problemen, die der Kommunikation zwischen den Teammitgliedern und darin enthaltenen Störungen zuzuordnen sind (vgl. GÖLDNER, 2005).

Im – neben Problemsituationen und Unterstützung – dritten Themenkomplex des *VitaminL*-Projekts sind Fragen nach der Teamzusammensetzung einzuordnen. So stellt sich prinzipiell die Frage nach einer optimalen Zusammensetzung virtueller Teams auf der Grundlage eines gegebenen Rollenmodells. Vorbereitend zur Klärung dieser zentralen Frage nimmt sich SCHILL (2007) in ihrer Arbeit der Frage an, inwiefern sich gute und schlechte Teams bereits während der Zusammenarbeit erkennen und voneinander unterscheiden lassen. In diesem Zusammenhang lassen sich weitere Untersuchungen definieren wie beispielsweise die Frage nach demjenigen Teammitglied, das bei einer gegebenen Teamzusammensetzung zu einer optimalen Ergänzung des Teams führen würde.

Kapitel 8

Technische Aspekte des VitaminL-Systems

Die in Kapitel 7.3.3 genannten Folgerungen lassen den Schluss zu, dass die in 7.1.2 skizzierte Lernsituation durch eine CSCL-Umgebung (resp. ein ITS) sinnvoll unterstützt werden kann und sollte. Zur Klärung der im Rahmen der vorliegenden Arbeit relevanten Forschungsfragen (Identifikation von Problemsituationen und Erkennung von Teamfunktionen während der synchronen Zusammenarbeit; s. Kap.7.3.4.1 und 7.3.4.2) sind an das einzusetzende Software-System spezielle Anforderungen zu stellen:

1. Unterstützungsfunktionen:
   Die Unterstützung von virtuellen Teams bei der synchronen Problemlösung innerhalb eines Lernprozesses stellt eine grundlegende Anforderung an das System dar. Insbesondere die Unterstützung von Kommunikation (beispielsweise in Form eines Chats) und Kooperation (durch gemeinsame Informationsobjekte; s. Kap.5.5.3) wird verlangt.

2. Anwendungsgebiet:
   Aus dem Anwendungskontext der objektorientierten Programmierung in Java ergeben sich weitere, spezifische Erfordernisse wie die Integration spezieller Werkzeuge der Programmentwicklung (u.a. in Form von Java-Compiler und Java-Interpreter).

3. Kosten:
   Aufgrund der Budgetsituation des VitaminL-Forschungsprojekts sind die Anschaffungskosten des Software-Systems zu minimieren. Freeware-Projekte (oder Eigenentwicklungen) sind daher zu bevorzugen.

4. Erweiterbarkeit:
   Da ein Ausbau des einzusetzenden (CSCL-)Systems zu einem ITS angestrebt ist, ist eine Weiterentwicklung – und damit verbunden der Zugriff auf den Quellcode – des Systems unabdingbar.

5. Kontrollierbarkeit:
   Die geplanten Analyseansätze erforden einen Zugriff auf sämtliche Aktionen (inklusive der Kommunikation), die während der Zusammenarbeit von den einzelnen Gruppenmitgliedern ausgelöst werden.

Zur bestmöglichen Erfüllung dieser Anforderungen, insbesondere der beiden letztgenannten Kriterien, wurde einer Eigenentwicklung der Vorzug gegenüber bereits existierenden Systemen wie beispielsweise Habipro (s. Kap.6.4.1) oder EPSILON (s. Kap.6.4.4) gegeben. Für das weitere Verständnis der vorliegenden Arbeit wichtige Schritte und Konzepte der Entwicklung der VitaminL-Software sind Gegenstand der nun folgenden Abschnitte dieses Kapitels.

8.1 Machbarkeitsstudie

Der eigentlichen Entwicklung vorausgehend wurden anhand eines Prototypen die prinzipielle Machbarkeit sowie die Akzeptanz eines Software-Werkzeugs zur verteilten, synchronen Programmierung überprüft. Die Entwicklung des Prototypen fand während des Sommersemesters 2003 statt, der erste Einsatz im Rahmen dieser Vorabstudie erfolgte im September desselben Jahres.

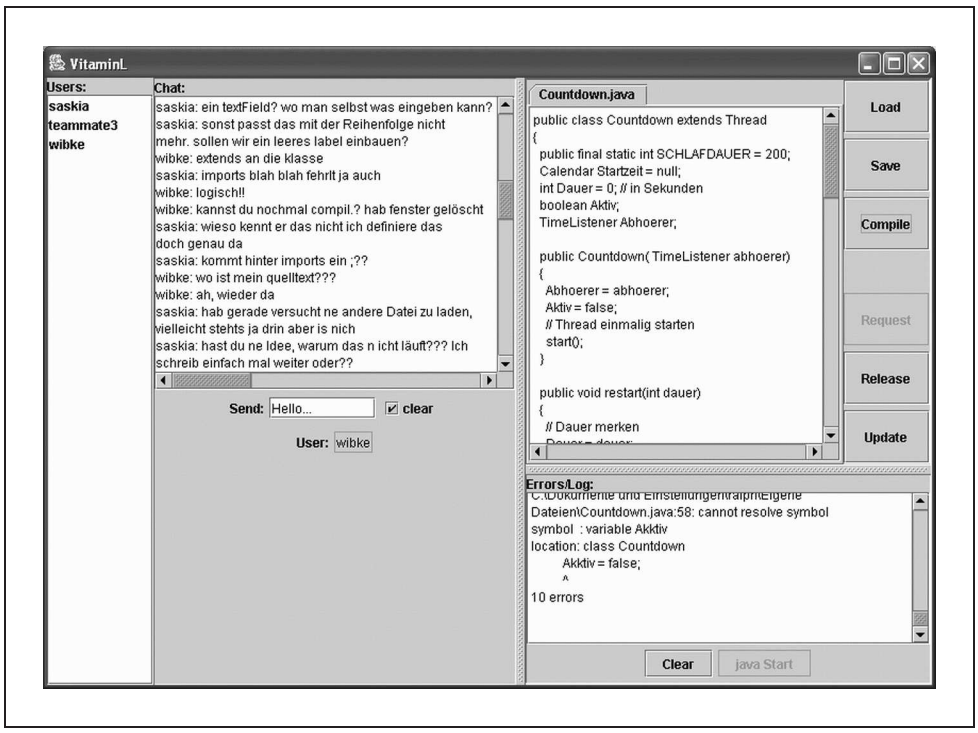

Abb. 8.1: Prototyp der VitaminL-IDE V1.0
(Screenshot)

Es folgt eine Erläuterung der für die Zusammenarbeit bei der Programmierung wesentlichen Konzepte.

8.1.1 Basiskonzepte

In der resultierenden Version 1.0 (s. Abb.8.1) konnten bereits wesentliche derjenigen Konzepte realisiert werden, die aus Sicht von CSCL für eine erfolgreiche Gruppenarbeit im Kontext der objektorientierten Java-Programmierung erforderlich sind.

8.1.1.1 Awareness

Die Unterstützung der gegenseitigen Wahrnehmung (s. Abschitt 4.6.1) erfolgt durch Anzeige der Namen aller aktuell angemeldeten Teilnehmer in einer Art Liste, die am linken Rand des Applikationsfensters positioniert ist. Der eigene Name des jeweiligen Benutzers erscheint zusätzlich in der unteren Hälfte des mittleren Fensterbereichs, welcher für die Kommunikation vorgesehen ist (s. Abb.8.2).

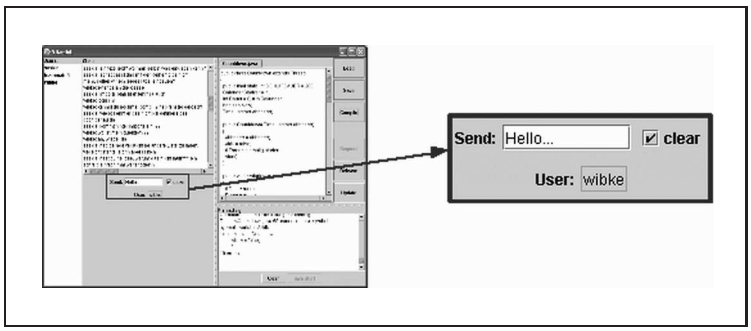

Abb. 8.2: Anzeige des Benutzernamens in der VitaminL-IDE V1.0
(Screenshot)

Diese Unterstützungsfunktion wurde in späteren Versionen geringfügig überarbeitet: Der Name des Benutzers erscheint dann in der Titelleiste der Applikation, die Benutzerliste wird um Benutzer-Icons erweitert und zusätzlich wird eine Liste aller verfügbaren sowie aktiven Gruppen dargestellt (s. Kap.8.4.1.2).

8.1.1.2 Kommunikation

Die Kommunikation wird in VitaminL in Form eines einfachen, text-basierten Chats (s. Kap.3.1.1) realisiert, wobei Versand und Empfang von Textnachrichten wie folgt umgesetzt wurden:

- Senden einer Textnachricht:
  In einem einzeiligen Textfeld (s. Abb.8.2) wird die zu sendende Nachricht eingegeben und – nach Betätigung der <CR>-Taste – an alle Gruppenmitglieder verschickt.

- Empfang einer Textnachricht:
  Sämtliche eingehenden Nachrichten werden in einem Textfenster (s. Abb.8.3) in der zeitlichen Reihenfolge des Empfangs untereinander ausgegeben, das

heißt neue Nachrichten werden an das untere Ende der listenartigen Ausgabe angehängt.

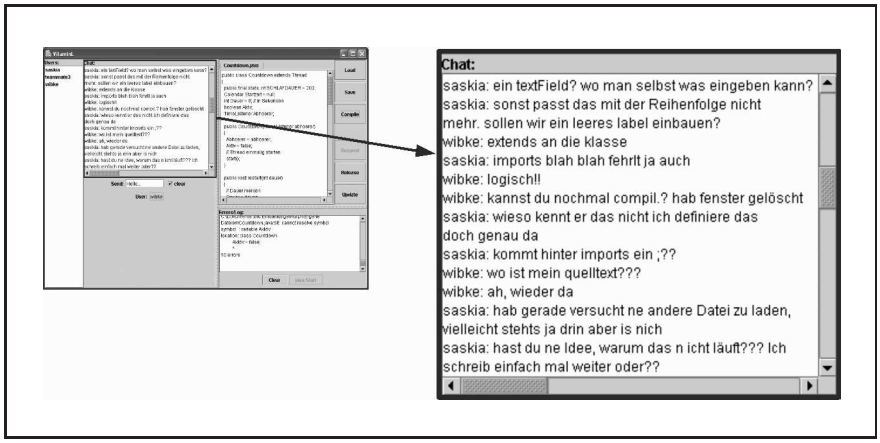

Abb. 8.3: Nachrichtenempfang in der VitaminL-IDE V1.0
(Screenshot)

In der Übersicht der eingegangenen Nachrichten können bei Bedarf ältere Nachrichten über entsprechende Bedienelemente (sog. *Scrollbars*; dt.: Schieberegler) gezielt ausgewählt und angezeigt werden. Somit ist während einer Sitzung stets die gesamte Kommunikationshistorie für alle Teilnehmer verfügbar und einsehbar. Die im VitaminL-System umgesetzte Kommunikationsstruktur (s. Kap.2.3.3) entspricht somit einem vollständig vermaschten Netz (s. Abb.2.4).

8.1.1.3 Koordination

Eine explizite Koordinierungsfunktion (beispielsweise mittels Gruppenkalender zur Terminplanung oder per To-Do-Listen zur Aufgabenverteilung) ist in VitaminL derzeit nicht vorhanden: Da VitaminL zunächst schwerpunktmäßig zur Unterstützung bei Tutorien und Hausaufgaben angedacht ist, nimmt eine Aufgabenplanung und - verteilung nur einen sehr geringen Anteil an der gesamten Bearbeitungsdauer ein. Im Bedarfsfall kann jedoch behelfsweise auf die Kommunikationshistorie zurückgegriffen werden.

8.1.1.4 Kooperation

Die Zusammenarbeit innerhalb der Gruppe erfolgt auf Basis eines Gruppeneditors (s. Kap.4.6.4.3), der in der rechten Hälfte des Anwendungsfensters zu finden ist und in dieser ersten Version die Bearbeitung eines einzelnen Dokuments unterstützt, dessen Inhalt über Lade- und Speicheroperationen entsprechenden Dateien zugeordnet werden kann. Ein grundlegendes Bedienkonzept des Gruppeneditors von VitaminL in dieser wie auch in nachfolgenden Versionen stellt die Metapher des Schreibstifts dar, welcher mit einem Dokument verknüpt ist: Dasjenige Mitglied, das eine Datei im Gruppeneditor (durch Laden von einem Datenträger) öffnet,

besitzt das alleinige Schreibrecht auf diesem Dokument und kann den enthaltenen Quelltext bearbeiten. Die übrigen Mitglieder besitzen nur Leserechte, können aber die Änderungen im Dokument mitverfolgen.

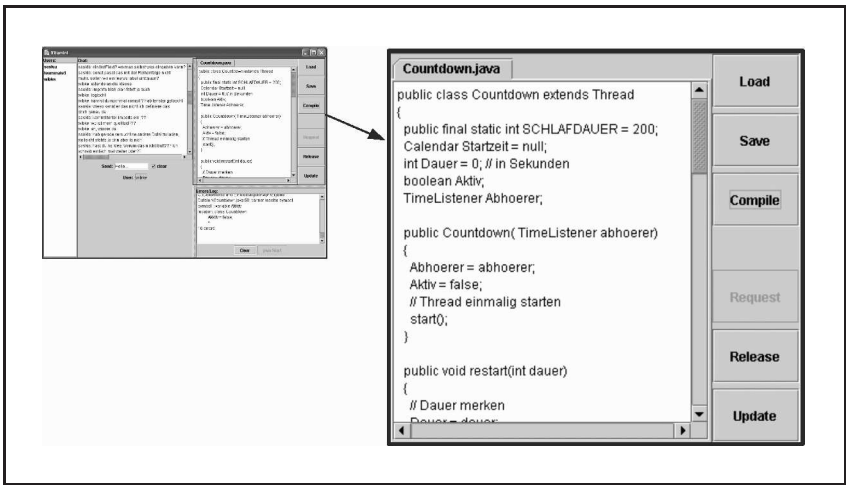

Abb. 8.4: Der Gruppeneditor in der VitaminL-IDE V1.0
(Screenshot)

Mittels entsprechender Bedienelemente kann der gedachte Stift – und damit das an ihn gebundene Schreibrecht – vom derzeitigen Dokumentbesitzer abgegeben werden (Schaltfläche *Release*) und anschließend von einem beliebigen Mitglied aufgenommen werden (*Request*). Auf diese Weise können – entsprechende Koordination innerhalb der Gruppe und ihrer Mitglieder vorausgesetzt – alle Gruppenmitglieder aktiv an den Quelltexten einer Programmieraufgabe mitarbeiten.

Ergänzt werden die eben genannten Kooperationswerkzeuge im Hinblick auf die Domäne objektorientierter Programmierung in Java durch die Integration eines Java-Compilers: Nach Auswahl der Funktion *Compile* wird das aktuelle Quelltext-Dokument dem Compiler zur Übersetzung übergeben, auftretende Fehlermeldungen werden in einem separaten Textbereich unterhalb des Dokuments ausgegeben.

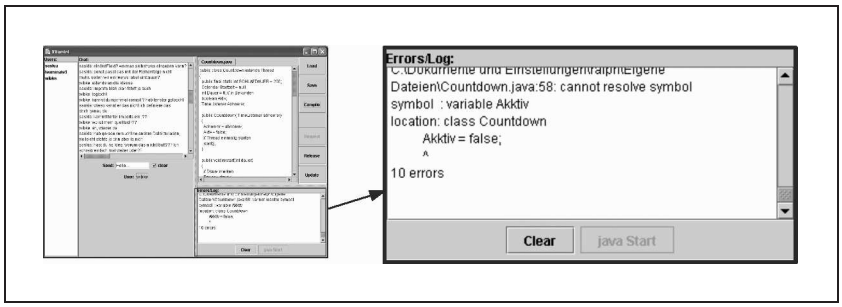

Abb. 8.5: Fehlermeldung des Java-Compilers in der VitaminL-IDE V1.0
(Screenshot)

Da auch diese (Fehler-)Meldungen allen Mitgliedern bekannt gemacht werden, ist eine gemeinschaftliche Korrektur fehlerbehafteter Quelltexte unter Zuhilfenahme von Chat und Gruppeneditor durchführbar. Da in der vorliegenden Version V1.0

noch kein Java-Interpreter integriert ist, muss zum Ausführen der Java-Programme zusätzlich zur VitaminL-Applikation eine Shell gestartet werden.

8.1.2 Durchführung von Benutzertests

Die Erprobung dieses Prototypen hinsichtlich prinzipieller Verwendbarkeit zur synchronen, verteilten Java-Programmierung einerseits und Akzeptanz seitens der Zielgruppe andererseits erfolgte durch Benutzertests, die erstmalig zwischen dem 5. September 2003 und dem 8. Oktober 2003 stattfanden.

8.1.2.1 Struktur der Teilnehmer

Insgesamt nahmen zehn Studierende in vier Gruppen, verteilt auf je zwei Gruppen mit zwei beziehungsweise drei Mitgliedern, an dieser ersten Studie teil. Alle Teilnehmer hatten bereits zwei Semester im Studiengang IIM an der Universität Hildesheim absolviert und konnten daher Java-Kenntnisse vorweisen, die sie im Sommersemester 2003 (also unmittelbar vor den Benutzertests) erworben hatten. An dieser Studie nahmen bewusst *keine* Anfänger in der Java-Programmierung teil, da der Fokus zunächst auf die prinzipielle Anwendbarkeit der VitaminL-Software gerichtet war. Diese Frage sollte geklärt werden, bevor die Klärung der eigentlichen Forschungsfragen behandelt werden sollte.

8.1.2.2 Beschreibung der Testsituation

Die durchgeführten Benutzertests fanden zunächst unter Laborbedingungen statt. Unter Laborbedingung soll hier verstanden werden, dass die eigentliche Zielsituation, in welcher die Teilnehmer als virtuelles Team zwar standortübergreifend, aber dennoch synchron gemeinsam Programmieraufgaben bearbeiten, modifiziert wird zugunsten einer besseren Kontrollierbarkeit seitens der Versuchsleitung. Konkret bedeutet dies, dass sich alle Teilnehmer während eines Benutzertests im selben Raum befinden, aber einerseits durch eine geeignete Anordnung der Arbeitsplätze und andererseits durch ein vorab vereinbartes Schweigegebot für die Dauer des Experiments künstlich voneinander isoliert werden, um auf diese Weise eine räumliche Trennung zu simulieren.

Mit dieser Verfahrensweise kann folgendes sichergestellt werden:

- Die Teilnehmer arbeiten ausschließlich unter Verwendung des zu erprobenden Systems miteinander. Insbesondere ein Ausweichen auf andere Kommunikationsmittel (wie Internet-Telefonie via Skype oder Instant Messenger) sowie eine Missachtung des vereinbarten Schweigegebots werden unterbunden.
- Die Versuchsleitung kann bei technischen Problemen, die insbesondere in einem frühen Entwicklungsstadium nahezu unvermeidbar sind, eingreifen, um einen möglichst reibungslosen Sitzungsablauf zu gewährleisten.

Jede Sitzung war auf zwei Stunden begrenzt und wurde wie in nachfolgender Tabelle 8.1 strukturiert.

Tab. 8.1: Strukturierung von Benutzertests

Phase	**Dauer**	**Inhalt**
Einführung	13 min	In der Einführungsphasen wird den Teilnehmern das Software-System vorgestellt, indem an einem praktischen Beispiel der Funktionsumfang des Prototypen demonstriert wird. Abschließend erhalten die Teilnehmer die Möglichekeit, noch offene Fragen hinsichtlich des Systems und seiner Bedienung zu klären.
Vorbereitung	2 min	Nach einem Neustart des Systems melden sich alle Teilnehmer an der VitaminL-Anwendung an. Zeitgleich wird die zu bearbeitende Aufgabe (in ausgedruckter Papierform) an alle Probanden verteilt. Auf diese Weise wird insgesamt sichergestellt, dass sowohl innerhalb einer Gruppe als auch unter den Gruppen identische Anfangsbedingungen im Hinblick auf die Arbeitsumgebung und die Programmieraufgabe herrschen.
Bearbeitung	100 min	In dieser Phase bearbeiten die Teilnehmer gemeinsam und synchron die gestellte Aufgabe unter ausschließlicher Verwendung der VitaminL-Anwendung.
Nachbereitung	5 min	Zum Abschluss eines Benutzertests findet eine kurze Diskussion über das in der Sitzung erarbeitete Ergebnis statt mit anschließender Präsentation eines Lösungsvorschlags durch die Versuchsleitung.

Im Anschluss an die jeweiligen Benutzertests erhielten alle Teilnehmer einen Fragebogen mit Fragen zum Experiment, zur Aufgabenbearbeitung und zur VitaminL-Applikation selbst. Die Antworten der Teilnehmer gaben einerseits Hinweise auf die Akzeptanz hinsichtlich des Einsatzes der VitaminL-IDE und enthielten andererseits auch Anhaltspunkte für mögliche beziehungsweise notwendige Verbesserungen und Erweiterungen des Prototypen.

8.1.3 Ergebnisse

Von den zehn an die Teilnehmer ausgeteilten Fragebögen wurden acht Stück ausgefüllt zurückgegeben und konnten zur Auswertung herangezogen werden. Dies entspricht einer Rücklaufquote von 80 %.

8.1.3.1 Der erste Eindruck

Nach dem ersten Eindruck befragt (*„Nennen Sie uns in einem kurzen Satz Ihren allerersten Eindruck vom soeben durchgeführten Experiment!"*) äußerten sich nahezu alle Teilnehmer positiv. So beurteilte ein Proband das Experiment *„als eine gute und sinnvolle Art, modern zusammen zu arbeiten bzw. zu kommunizieren, auch wenn es hier und da noch Verbesserungen geben kann."* Eine Teilnehmerin lobte *„eine ganz neue Arbeitsweise"*, die ihr *„viel Spaß gemacht"* hat. Andere Teilnehmer bezeichneten das Experiment als *„innovativ und durchaus sinnvoll"*, als *„interessant"* und auch als *„durchaus gut"*.

Getrübt wurde dieser durchaus positive erste Eindruck durch einige technische Probleme, die sich zum Teil in langen Wartezeiten (beispielsweise während der Kommunikation oder auch beim Aufruf des integrierten Java-Compilers) oder auch vereinzelten Abstürzen des Programms zeigten. Hier machte sich der prototpyische Charakter der allerersten Version von VitaminL bemerkbar. Entsprechend beurteilte eine Teilnehmerin das System als *„noch nicht ganz ausgereift"*, andere *„wurden aber leider ausgebremst"*, eine weitere Teilnehmerin fasste dann auch zusammen: *„Das Experiment war nicht schlecht, nur die Fehler, die aufgetreten sind, haben etwas die Arbeit ziemlich beeinträchtigt."*

8.1.3.2 Aufgetretene Probleme

Auf die Frage, *„Welche Probleme sind während der Zusammenarbeit aufgetreten?"*, standen überraschenderweise weniger die offensichtlichen, technischen Probleme im Vordergrund: Lediglich eine Teilnehmerin kritisierte, *„der Computer ist mehrmals abgestürzt und das Compilieren hat mehrere Minuten gedauert, was die Arbeit sehr beeinträchtigt hat."* Eine weitere Probandin beklagte *„langsame Ladezeiten"*. Die Mehrheit der Teilnehmer bemängelte die noch unausgereiften technischen Fähigkeiten des Gruppeneditors: Da in der vorliegenden Version lediglich ein Dokument gleichzeitig bearbeitet werden konnte und nur ein Anwender die entsprechenden Schreibrechte (zu einem Zeitpunkt) besitzen konnte, hatten alle anderen Teammitglieder automatisch eine Art Beobachterstatus mit der Möglichkeit, per Chat zu kommunizieren und auf diese Weise ihre Beiträge zur Zusammenarbeit zu leisten. Dies jedoch wurde durch den Gruppeneditor erschwert, denn man *„wusste nie genau, wo der Teamkollege gerade war / schaute"*, was teilweise dazu führte, *„dass nur einer wirklich arbeit und der eigentlich auf sich alleine gestellt ist"*. So – oder ähnlich – äußerten sich immerhin vier der Teilnehmer. Als weitere Schwachpunkte des Editors wurde unter anderem auf fehlende Zeilennummern hingewiesen, infolgedessen die *„Verständigung bzgl. Zeilenangabe"* erschwert wurde, sowie auf die fehlende *„Möglichkeit, Zeilen zu kopieren und einzufügen über die rechte Maustaste"*.

Probleme bei der Java-Programmierung wie in Kapitel 7.2.2 wurden wider Erwarten gar nicht genannt. Dies lässt sich jedoch anhand der Teilnehmerstruktur dieser Studie erklären, handelt es sich bei diesen doch ausschließlich um solche

Studierende, die zuvor eine vollständige Lehrveranstaltung zur objektorientierten Programmierung in Java besucht und zu dem Zeitpunkt der Studie bereits ihre Abschlussprojekte bearbeitet hatten.

8.1.3.3 Negative Aspekte

Ergänzend zu den Problemen wurden die Teilnehmer auch nach negativen Aspekten der Zusammenarbeit gefragt: *„Was hat Ihnen an dieser Art der Zusammenarbeit nicht gefallen?"* Daraufhin kritisierten drei Teilnehmer den im Vergleich zur konventionellen Gruppenarbeit (s. Kap.7.1.2) höheren Zeitaufwand. So bemerkte eine der Teilnehmerinnen recht knapp, *„real hätten wir halb so lang gebraucht"*, während eine zweite Probandin diesen Umstand etwas ausführlicher begründete: *„Manchmal dauert es lange, bis die Reaktion des anderen kommt, weil ja immer erst getippt werden muss. Außerdem muss man überlegen, wie man das, was man sagen möchte, knapp, präzise und verständlich schreibt. Das dauert einfach länger."*

Zwei weitere Teilnehmer antworteten auf die Frage, *„dass nur einer wirklich arbeitet und der eigentlich auf sich alleine gestellt ist"*, was zur Folge hatte, dass die übrigen Teammitglieder *„überwiegend nur Beobachter der Aktionen"* gewesen sind. Kurioserweise sah eine Teilnehmerin genau dies in einem anderen Licht und hob positiv hervor, *„dass jeweils nur einer schreibt, es aber alle sehen können, ist gut."* Als weitere Krititkpunkte wurden eine fehlende Unterstützung bei der Aufgabenverteilung und das Design des Programms (*„Die Oberfläche war teilweise nicht so benutzerfreundlich."*) genannt.

8.1.3.4 Positive Aspekte

Den genannten Problemen stehen auch positive Aspekte gegenüber. Auf die Frage, *„was hat Ihnen an dieser Art der Zusammenarbeit gefallen?"*, stellten alle Teilnehmer die Möglichkeit, standortübergreifend als virtuelles Team zusammenzuarbeiten, besonders positiv hervor. Einer Teilnehmer gefiel es beispielsweise, *„dass man per Computer zusammenarbeiten konnte und bei Fragen die anderen fragen konnte"*, ein anderer Proband betonte *„die Verbindung von Chat und Arbeitsprogramm"*, die übrigen Teilnehmer äußerten sich in vergleichbarer Weise. Eine Studentin fasste das Experiment treffend zusammen als *„eine sehr modernde Art, gemeinsam an einem Projekt zu arbeiten."*

8.1.4 Folgerungen

Zunächst darf festgestellt werden, dass virtuellen Teams prinzipiell ein gemeinschaftliches Programmieren in Java mit der VitaminL-IDE ermöglicht wird. Die im Rahmen der durchgeführten Studie erzielten Arbeitsergebnisse lassen diesen Schluss zu – trotz aller Probleme, die während der Sitzungen aufgetreten sind. Diese lassen sich einerseits auf den Entwicklungsstatus der VitaminL-Software

zuückführen, andererseits aber auch auf die für alle Teilnehmer ungewohnte Lernsituation. So urteilte eine Probandin auch zusammenfassend *„wenn man mit den selben Leuten öfter so zusammenarbeitet, ist es sicher sehr effektiv."*

Unter Einbeziehung der erreichten Arbeitsergebnisse überwiegen die positiven Eindrücke, Meinungen und Aspekte, so dass mit dieser Studie – neben einer reinen Machbarkeit – auch die Akzeptanz des skizzierten Arbeits- und Lernansatzes seitens der Zielgruppe sichergestellt werden konnte.

In der Folge wurde der Prototyp weiterentwickelt und sukzessive zu einer Applikation ausgebaut, die das gemeinsame Programmieren unterstützen und den dabei stattfindenden Lernprozess fördern soll. Dabei liegen die Schwerpunkte der Folgeversion in der Unterstützung der Kommunikation mittels einer strukturierten Dialogschnittstelle (s. Kap.3.1.4) sowie im Ausbau des Gruppeneditors, um echtes Document Sharing, das heißt das gleichzeitige Bearbeiten mehrerer Dokumente durch mehrere Anwender, zu ermöglichen.

8.2 Strukturierte Komunikation

8.2.1 Motivation

Wie mit der vorigen Studie belegt werden konnte, ist text-basierter Chat prinzipiell für kommunikative Zwecke im Hinblick auf eine erfolgreiche Bearbeitung von Programmieraufgaben durch virtuelle Teams einsetzbar. Da hinsichtlich der in Kapitel 7.3.4 erläuterten Fragestellungen auch der Inhalt der Kommunikation – neben weiteren für die Zusammenarbeit relevanten Aktionen – wichtig ist, wird ein Ansatz benötigt, die Kommunikation während der Zusammenarbeit analysieren zu können.

Anstelle eines Parsers zur Erkennung natürlicher Sprache soll in VitaminL ein anderer Analyseansatz Verwendung finden, der sich weniger am eigentlichen Inhalt, sondern mehr an der Intention von Äußerungen orientiert. Für diesen Ansatz wird auf die *Collaborative Learning Skills* (s. Kap.3.1.4) zurückgegriffen, welche im Rahmen des Projekts EPSILON (s. Kap.6.4.4) bereits erfolgreich im Kontext objektorientierter Software-Entwicklung zur Anwendung kommen.

Durch die systematische Kategorisierung kommunikativer Äußerungen (in *skills*, *subskills* und *attributes*) ergibt sich eine hierarchische Zerlegung mit einer Baumstruktur, in welcher jedem Blattknoten ein kommunikativer Akt mit einer bestimmten Absicht zugeordnet ist. Darüber hinaus wird jeder dieser Kommunikationsakte auf eine passende Phrase beziehungsweise einen zutreffenden Satzanfang abgebildet.

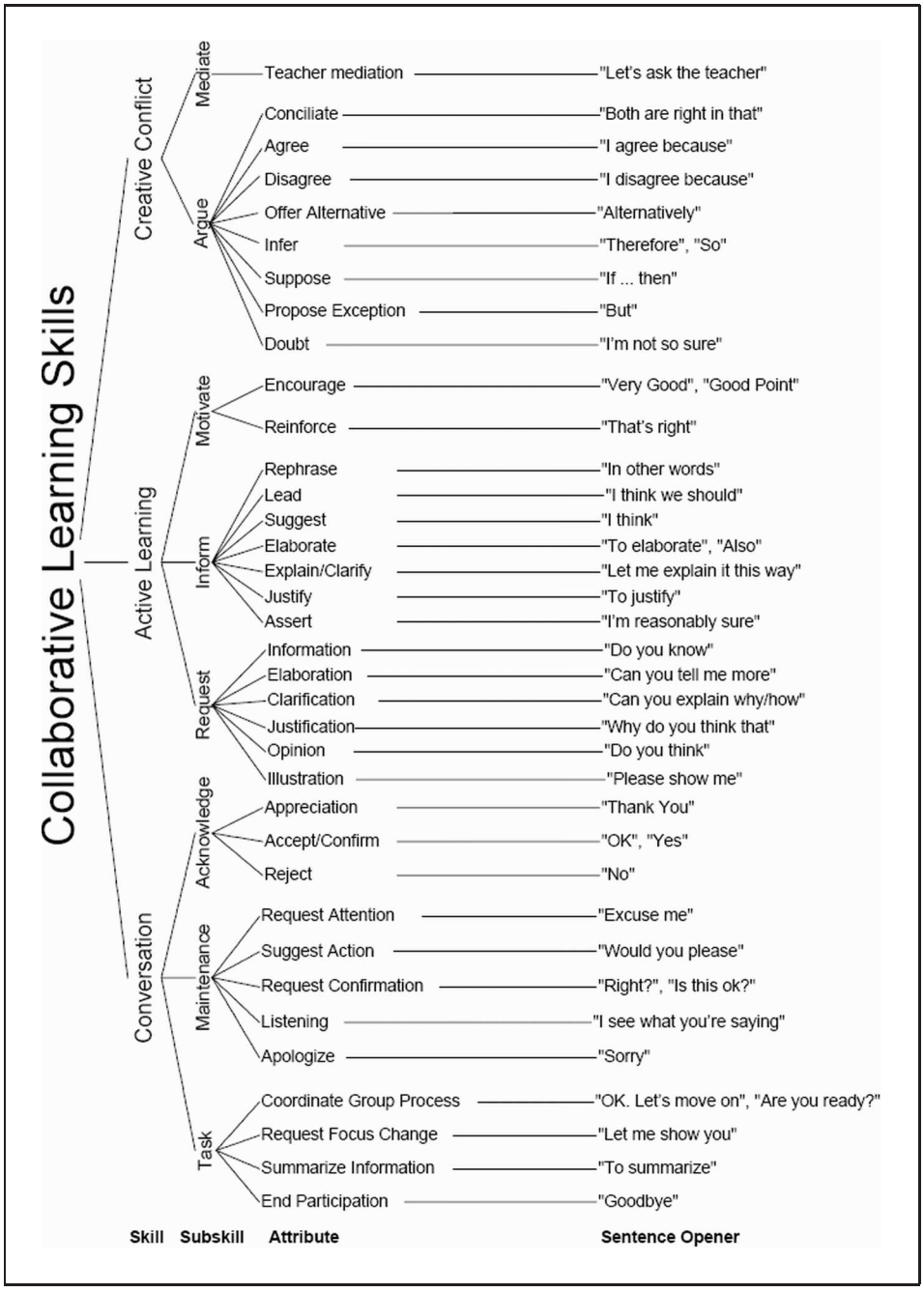

Abb. 8.6: Taxonomie der *Collaborative Learning Skills* (CLS)
(Soller, 2001, S.7)

Für das VitaminL-Projekt wurde die ursprüngliche Taxonomie des EPSILON-Projekts (s. Abb.8.6) in die deutsche Sprache übersetzt und mittels einer einfachen Numerierung codiert. In Tabelle 8.2 ist das Resultat zu sehen. Das Ergebnis stellt eine spezielle Art der Codierung der gesamten Kommunikation dar, in Folge derer auf eine weitere inhaltliche Analyse von Botschaften verzichtet wird.

Tab. 8.2: Abbildung der CLS-Attribute auf Satzanfänge und Codes

Attribut	**Satz(anfang)**	**Code**
Koordination	Okay, lasst uns fortfahren.	0
Themenwechsel	Lasst mich Euch zeigen...	1
Zusammenfassung	Zusammenfassend...	2
Anerkennen	Danke schön.	3
Akzeptieren	Ja.	4
Ablehnen	Nein.	5
Entschuldigen	Verzeihung.	6
Zuhören,Verstehen	Ich sehe, was Du sagen willst...	7
Zustimmung einholen	Richtig?	8
Tätigkeit vorschlagen	Würdest Du bitte...?	9
Aufmerksamkeit erbitten	Entschuldigt mich...	10
Information	Weisst Du...?	11
Einzelheiten	Kannst Du mir mehr sagen...?	12
Klärung	Kannst Du erläutern, warum/wie...?	13
Rechtfertigung	Warum denkst Du, dass...?	14
Meinung	Denkst Du...?	15
Erklärung	Bitte zeige mir...	16
Schlichtung	Beide haben recht...	17
Zustimmung	Ich stimme zu, weil...	18
Widerspruch	Ich stimme nicht zu, weil...	19
Alternativangebot	Alternativ dazu...	20
Folgerung	Deshalb...	21
Annahme	Wenn..., dann...	22
Ausnahme	Aber...	23
Zweifel	Ich bin nicht sicher...	24
Neuformulierung	Mit anderen Worten...	25
Führung	Ich denke, wir sollten...	26
Vorschlag	Ich denke...	27
Ausarbeitung	Zur Ausarbeitung...	28
Erklärung	Lasst es mich so erklären...	29
Rechtfertigung	Zur Rechtfertigung...	30
Behauptung	Ich bin ziemlich sicher...	31
Ermutigung	Sehr gut.	32
Verstärkung	Das ist richtig.	33
Hilfe durch Tutor	Lasst uns den Tutor fragen.	34

8.2.2 Prototypische Umsetzung strukturierter Kommunikation

In Anlehnung an die Dialogschnittstelle von EPSILON (s. Abb.6.11, untere Hälfte) wurde für den VitaminL-Prototypen eine GUI-Komponente gestaltet, die den Versand von Nachrichten, basierend auf den CLS, realisiert.

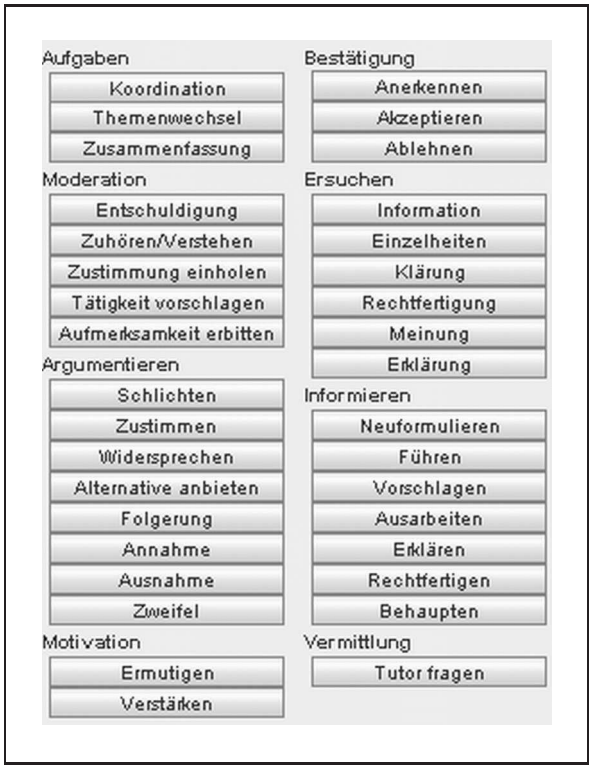

Abb. 8.7: Prototyp der strukturierten Dialogschnittstelle von VitaminL (Screenshot)

Wie in der Abbildung 8.7 zu sehen ist, werden die CLS-Attribute über Schaltflächen realisiert und in Gruppen jeweils unterhalb der zugehörigen Kategorie (d.h. auf Ebene der *subskills*) angeordnet. Nach Auswahl eines CLS-Attributs durch Betätigung der entsprechenden Schaltfläche wird ein Dialogfenster geöffnet, in welchem der zum gewählten Attribut gehörige Satzanfang angezeigt wird (s. Abb.8.8).

Abb. 8.8: Nutzung strukturierter Kommunikation: Satzanfang (Screenshot)

In einem Textfeld kann der ausgewählte Satzanfang vom Benutzer durch entsprechende Texteingaben vervollständigt werden und wird – nach Betätigung der Schaltfläche *„Ende"* – zusammen mit der Benutzereingabe als Nachricht an alle derzeit angemeldeten Teammitglieder gesendet (s. Abb.8.9).

Abb. 8.9: Nutzung strukturierter Kommunikation: Vervollständigter Satz (Screenshot)

Alle empfangenen Nachrichten werden unter Angabe des Absenders als Text in einem mit *„Kommunikation"* betitelten Teilbereich des Applikationsfenster angezeigt (s. Abb.8.10).

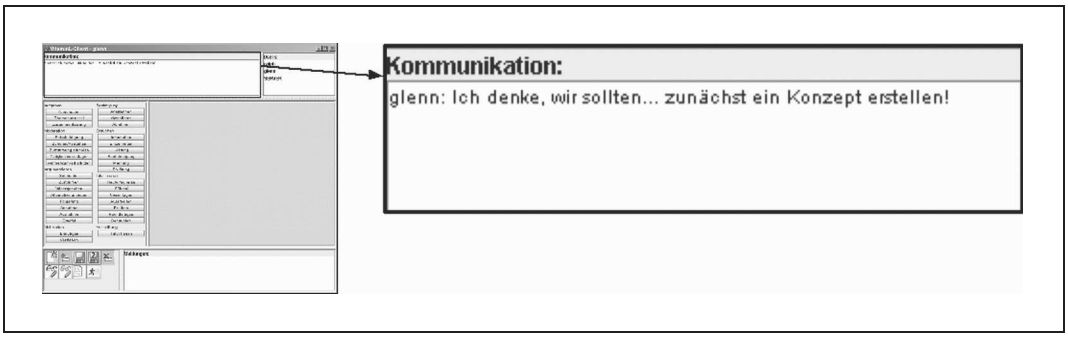

Abb. 8.10: Empfang einer Kommunikationsbotschaft (Screenshot)

Falls einem Attribut bereits ein vollständiger Satz zugeordnet ist (wie beispielsweise der Satz *„Ja."* dem Attribut *Akzeptieren*; s. Tab.8.2), wird dieser Satz direkt versendet, ohne den erwähnten Dialog anzuzeigen.

8.2.3 Ergebnisse bei Verwendung strukturierter Kommunikation

Die so gestaltete Kommunikationsschnittstelle wurde im Rahmen von weiteren Benutzertests, die vom 23.06.2004 bis zum 14.09.2004 stattfanden, eingesetzt und auf ihre Tauglichkeit hin überprüft. An den Sitzungen in diesem Zeitraum nahmen 27 Studierende in zwei Zweier-, fünf Dreier- und zwei Vierergruppen teil. Auch diese Teilnehmer hatten bereits an einer Veranstaltung zur Programmierung in Java teilgenommen und können – wie auch die Teilnehmer der Machbarkeitsstudie (s. Kap.7.2) – nicht mehr als Programmieranfänger bezeichnet werden.

Von den 27 Teilnehmern wurden 23 ausgefüllte Fragebögen zurückgegeben, die zur Auswertung herangezogen werden konnten. Die so gewonnenen Ergebnisse lassen zunächst keine eindeutige Interpretation zu.

1. Aus Sicht der Anwender erscheint die Kommunikation auf der Basis vorgegebener Sätze und Satzanfänge als *„ungewohnte und zeitaufwendige Kommunikation"*, auch bewerteten die Teilnehmer *„die Möglichkeiten der Kommunikation [als] zu eingeschränkt"* und kritisieren, dass *„die Kommunikationsanfänge meistens nicht passend"* waren. Als Folge davon sind bei einigen Teilnehmern *„Missverständnisse aufgetreten beim Chatten, da die Satzanfänge vorformuliert sind und das Problem teilweise damit nicht genau formuliert werden konnte."* So oder ähnlich kritisch äußerte sich nach dieser Testphase die überwiegende Mehrheit der Probanden[1], wobei einer der Teilnehmer zwar bemängelte, dass *„kein freier Chat"* verfügbar war und es *„bei der Kommunikation ... teilweise lange gedauert [hat], bis man den richtigen Button gefunden hat"*, jedoch gleichzeitig die *„praktische Kommunikation"* positiv hervorhob. Zusammenfassend lassen sich vier Äußerungen der Probanden hinsichtlich der Kommunikation mit der strukturierten Dialogschnittstelle des VitaminL-Prototypen als neutral beziehungsweise wertfrei einstufen, ein Kommentar ist positiver Natur und 15 Bemerkungen müssen negativ bewertet werden.

2. Aus dem Blickwinkel der Versuchsleitung betrachtet stellt die Verwendung einer strukturierten Dialogschnittstelle – trotz der genannten Kritik seitens der Anwender – einen durchaus praktikablen Ansatz zur zielgerichteten Kommunikation im Rahmen einer Programmieraufgabe dar. Dies spiegelt sich in den von den teilnehmenden Gruppen während der Sitzungen erarbeiteten Ergebnissen wieder. Darüber hinaus bietet die aus der Kategorisierung kommunikativer Akte resultierende Möglichkeit der Codierung eine interessante Alternative für eine Analyse der Kommunikation sowohl für Zwecke der Erkennung von Teamzusammensetzungen als auch im Rahmen der Identifikation potentieller Problemsituationen.

Um auch die Akzeptanz einer solchen Kommunikationsschnittstelle seitens der Anwender herzustellen, wurde ein Maßnahmenkatalog definiert, der bei nachfolgenden Benutzertests zur Anwendung gelangt. Die enthaltenen Maßnahmen sind sowohl technischer wie auch organisatorischer Art:

1. Aufklärung:
   In der Einführungsphase zu Beginn eines Benutzertests (s. Tab.8.1) werden die Teilnehmer explizit auf die strukturierte Dialogschnittstelle und daraus scheinbar resultierende Restriktionen bezüglich der Kommunikationsmöglichkeiten hingewiesen. In diesem Zusammenhang werden auch Sinn und Zweck dieser Kommunikationsform eingehend erläutert.

[1] Genau genommen kritisierten mit 19 von 23 Teilnehmern 82,6 % der Probanden in irgendeiner Form die Kommunikation per strukturierter Dialogschnittstelle

2. Re-Design:
   Die Bedienoberfläche der VitaminL-Applikation – und insbesondere auch der strukturierten Dialogschnittstelle – wurde komplett überarbeit, um software-ergonomischen Aspekten (wie beispielsweise besserer Bedienbarkeit und höherer Übersichtlichkeit) gerecht zu werden. Die wesentlichen Änderungen und Erweiterungen der Kommunikationskomponente werden im folgenden Kapitel 8.2.4 näher erläutert.

3. Bedienungsanleitung:
   Zwecks besserer Orientierungsmöglichkeiten bei der Benutzung der strukturierten Dialogschnittstelle wurde allen Teilnehmern zusammen mit der Aufgabenstellung eine ausgedruckte Übersicht über die Bedienung der (überarbeiteten) Dialogschnittstelle und alle zu den CLS-Attributen gehörenden Sätze (bzw. Satzanfänge) ausgehändigt (s. Abb.8.11).

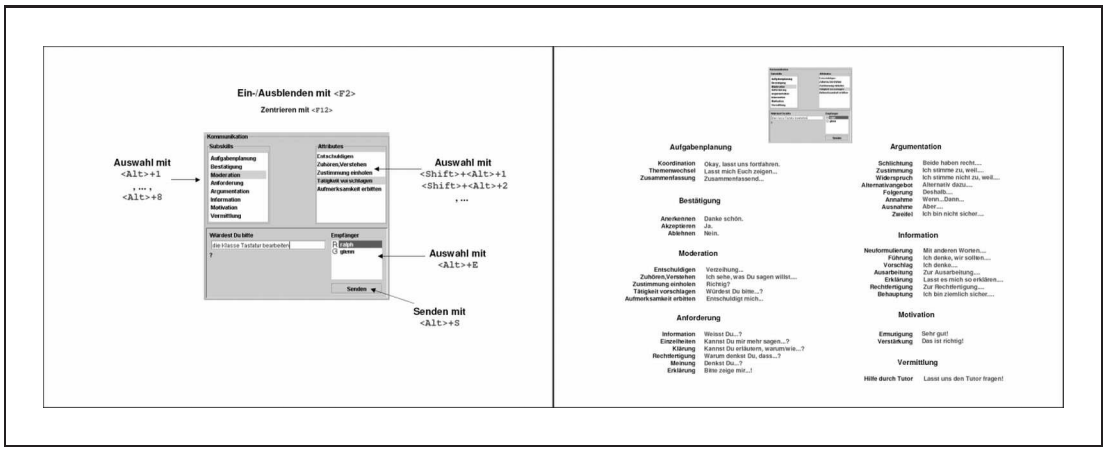

Abb. 8.11: Bedienmöglichkeiten der überarbeiteten Dialogschnittstelle (Handout)

8.2.4 Überarbeitete Kommunikationsschnittstelle

Ein wichtiger Punkt bei der Neugestaltung der Dialogschnittstelle war eine sinnvolle Nutzung der auf einem Bildschirm verfügbaren Anzeigefläche mit dem Ziel, den Anwendern der VitaminL-Software möglichst viel Platz für den Gruppeneditor *und* die Kommunikation bereit zu stellen. Aus diesem Grund wurde die Komponente zum Verfassen und Senden von Textbotschaften als – innerhalb der VitaminL-Applikation – eigenständiges Fenster definiert, das frei positioniert und bei Bedarf aus- und eingeblendet werden kann.

Auch inhaltlich fand eine komplette Überarbeitung statt: Anstelle von je einer Schaltfläche pro CLS-Attribut für den Direktzugriff auf die einzelnen Attribute erfolgt die Auswahl des gewünschten Attributs über Listenelemente. Die Vorgehensweise beim Verfassen und Absenden einer Textbotschaft ist nun wie folgt gestaltet:

1. Wahl einer Kategorie:
   Zunächst wählt der Anwender aus einer Liste die gewünschte Kategorie (*subskill*) aus (s. Abb.8.12(a)). Daraufhin wird die Liste für die Attributwahl aktualisiert. Es werden alle zur Kategorie gehörenden Attribute angezeigt.

2. Wahl eines Attributs:
   Nach der Auswahl des gewünschten Attributs wird der zugehörige Satz angezeigt (s. Abb.8.12(b)). Bei denjenigen Kommunikationselementen, die lediglich in Form eines Satzanfangs definiert sind, wird dieser Satzanfang mit entsprechenden Textfeldern für die Benutzereingaben zur Vervollständigung der Satzanfänge angezeigt.

3. Wahl eines dedizierten Empfängers:
   Optional kann aus einer dritten Liste ein Gruppenmitglied als *dedizierter Empfänger* einer Textbotschaft ausgewählt werden (s. Abb.8.12(c)). Die Nachricht wird dann weiterhin an *alle* Teammitglieder geschickt und kann dort gelesen werden. Zusätzlich wird die empfangene Nachricht bei dem gewählten Teammitglied optisch hervorgehoben und es wird ein akustisches Signal ausgelöst. Auf diese Weise kann während der Zusammenarbeit bei Bedarf gezielt die Aufmerksamkeit auf die Kommunikation gerichtet werden.

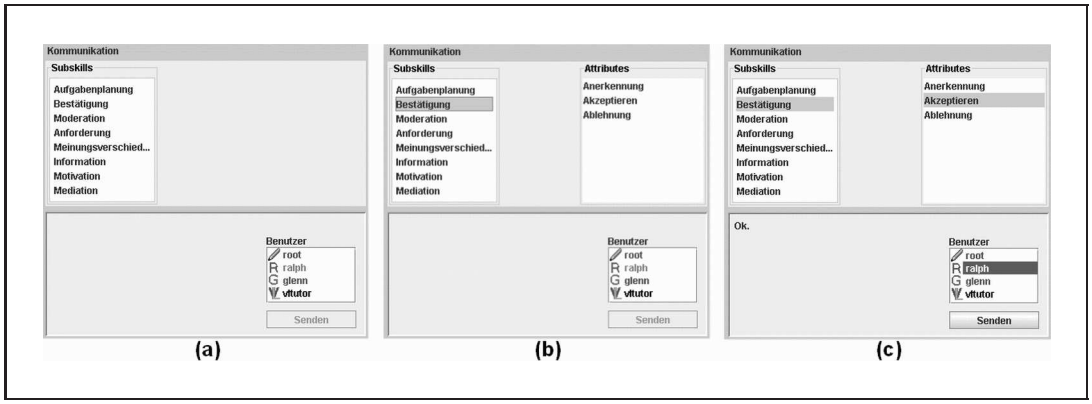

Abb. 8.12: Überarbeitete Dialogschnittstelle
(Screenshot)

Die überarbeitete Version der Dialogschnittstelle konnte im Rahmen einer dritten Phase von Benutzertests, die zwischen Dezember 2004 und März 2005 stattfanden, erstmalig eingesetzt werden.

8.2.5 Ergebnisse

An den Benutzertests mit der überarbeiteten Dialogschnittstelle nahmen 31 Studierende (überwiegend Erstsemester, die zu diesem Zeitpunkt eine Java-Programmierveranstaltung besuchten), verteilt auf elf Gruppen, teil. Von diesen 31 Teilnehmern gaben anschließend 21 einen ausgefüllten Fragebogen zur Auswertung zurück. Von diesen 21 Teilnehmern äußerten sich 18 zur Dialogschnittstelle.

Die Resonanz der Teilnehmer auf die überarbeitete Form der Kommunikation ist weitaus positiver zu bewerten verglichen mit dem Prototypen. Zwar existierten auch in dieser Tetphase kritische Stimmen seitens der Probanden, weil es *„zunächst schwierig [war], sich dieser Art der Kommunikation anzupassen"* (ähnlich formulierte es ein anderer Teilnehmer: *„Nur die Kommunikationsmöglichkeiten sind ein wenig zu dürftig und schränken ein wenig ein."*), daneben gab es jedoch auch eine Vielzahl positiver Äußerungen, die einer der Probanden treffend auf den Punkt bringt: *„Ich finde, der Chat reicht völlig aus. Es ist zwar erst gewöhnungsbedürftig, aber wenn man sich eingearbeitet hat, findet man sich ziemlich gut zurecht."*

Zusammenfassend ergeben sich sechs positive, vier neutrale und acht negativ zu bewertende Aussagen seitens der Teilnehmer hinsichtlich der Verwendung von strukturierter Kommunikation. Stellt man diesen Ergebnissen die Resultate der ersten Version einer strukturierten Dialogschnittstelle gegenüber, so kann eine deutliche Verbesserung der Akzeptanz seitens der Teilnehmer festgestellt werden.

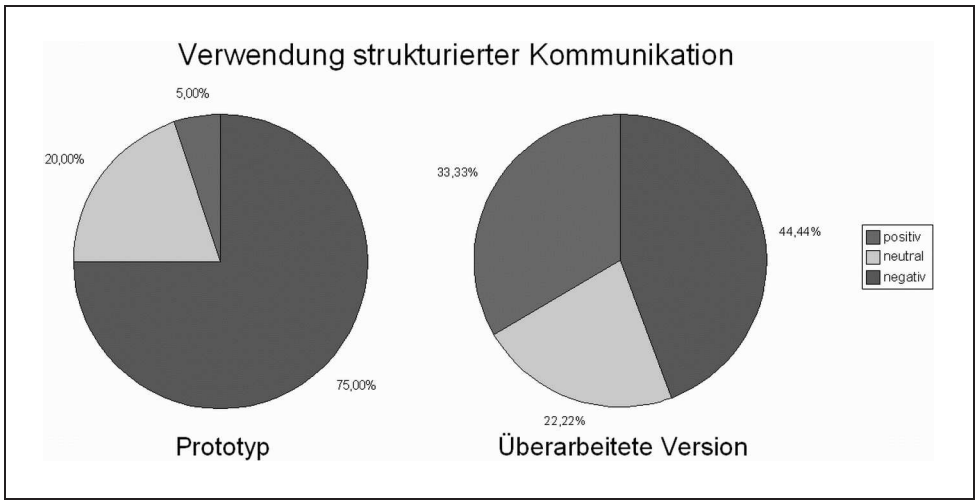

Abb. 8.13: Gegenüberstellung von Dialogschnittstellen

Wie in Abbildung 8.13 zu erkennen ist, hat sich die positive Resonanz von 5 % (einer von 20 Befragten) auf 33,3 % (sechs von 18 Befragten) erhöht, während die negative Kritik von 75 % (15 von 20 Befragten) auf 44,44 % (acht von 18 Befragten) reduziert werden konnte.

Da die Studierenden während der Sitzungen die gestellten Programmieraufgaben mit beiden hier vorgestellten Dialogschnittstellen hinreichende bearbeiten konnten, stellt sich die Frage, ob eine strukturierte Kommunikation zur Unterstützung bei der Programmierung Verwendung finden kann, nicht. Für die Akzeptanz seitens der Anwender ist es jedoch wichtig, geeignete Maßnahmen, sowohl hinsichtlich der Information der Anwender als auch bezüglich der Gestaltung der Dialogschnittstelle, durchzuführen.

8.3 Verteilte Dokumente

Die Kooperationsunterstützung in Form eines Gruppeneditors (s. Kap.4.6.4.3) stellt ein zentrales Bedienkonzept beim Programmierenlernen mittels der VitaminL-IDE dar. Der im Prototypen von VitaminL getestete Ansatz (s. Kap.8.1.1.3) erwies sich als prinzipiell nutzbar, ermöglichte jedoch nur einem Mitglied eines Programmierteams zu einem Zeitpunkt die aktive Arbeit an einem Dokument und versetzte alle anderen Mitglieder in einen Beobachterstatus. Zur Behebung dieses Mangels und damit einhergehend zur Verbesserung der Kooperationsmöglichkeiten wurde auch der Gruppeneditor einer Neugestaltung und Weiterentwicklung unterzogen.

8.3.1 Anforderungen an einen Gruppeneditor

Die synchrone, gemeinsame Bearbeitung von Quelltexten stellt ein wesentliches Element der in Kapitel 7.1 skizzierten Lernsituation dar. Dementsprechend muss der Gruppeneditor von VitaminL allen Teilnehmern den Lese- und Schreibzugriff auf alle für die jeweilige Aufgabe relevanten Dokumente ermöglichen. Im Idealfall hieße dies, dass alle Mitglieder zeitgleich Vollzugriff auf sämtliche geöffneten Dokumente hätten, was jedoch mit besonderen Problemen hinsichtlich der Synchronisation der Dokumente respektive ihrer Inhalte verbunden ist. Man denke nur an eine Situation, in der Teilnehmer X die Zeile n eines Dokuments D löscht, während zur gleichen Zeit Teilnehmer Y in derselben Zeile desselben Dokuments eine beliebige Korrektur durchführt.

Zur Vermeidung solcher und ähnlicher Synchronisationsprobleme wird der bereits erfolgreich erprobte Gruppeneditor auf eine beliebige Anzahl gleichzeitig zu bearbeitender Dokumente erweitert, wobei die Metapher des Schreibstifts auf jedes Dokument einzeln angewendet wird. Konkret bedeutet dies, dass jedes Teammitglied zu einem Zeitpunkt Besitzer eines oder mehererer Dokumente sein kann und entsprechende Schreibrechte auf allen seinen Dokumenten besitzt. Alle anderen Mitglieder besitzen weiterhin Lesezugriff auf diese Dokumente, können ihrerseits jedoch eigene Dokumente (mit entsprechenden Schreibrechten) geöffnet halten und bearbeiten. Einschränkend – und damit zur Vermeidung von Synchronisationsproblemen der genannten Art – darf jedes Dokument zu einem Zeitpunkt nicht mehrmals von verschiedenen Benutzern geöffnet seind. Dadurch kann der exklusive Schreibzugriff durch höchstens ein Mitglied gewährleistet werden. Über die Möglichkeiten, eigene Dokumente freizugeben und freigegebene Dokumente anzufordern, wird das gemeinsame Bearbeiten von Dokumenten – und insbesondere das Bearbeiten einzelner Quelltexte durch alle Mitglieder einer Lerngruppe – wie zuvor gefordert realisiert.

Dieses in VitaminL umgesetzte Konzept von Shared Documents wird ergänzt um spezielle Informationsfunktionen, mit denen der in früheren Benutzertests bemängelten Unübersichtlichkeit (s. Kap.8.1.3.2) entgegengetreten wird:

- Dokumentenbesitzer:
  Jedem Anwender ist ein kleines Graphiksymbol (Icon) zugeordnet, das in der Übersicht der angemeldeten Benutzer dem jeweiligen Namen vorangestellt ist. Dasselbe Symbol findet sich bei allen Dokumenten (dem Dokumentennamen vorangestellt) wieder, die derzeit dem jeweiligen Benutzer zugeordnet sind. Dokumente ohne Symbol haben aktuell keinen Besitzer (freigegebene Dokumente).

- Zeilennummern:
  Als Orientierungshilfe werden von demjenigen Dokument, das aktuell ausgewählt ist, die Zeilennummern des Dokumentenbesitzers und des Dokumentenbetrachters angezeigt[2] . Auf diese Weise wird der Betrachter eines Dokuments effizient darüber informiert, an welcher Stelle der Dokumentenbesitzer im gewählten Dokument zuletzt aktiv war.

- Markierungen:
  Falls der Besitzer eines Dokuments einen Bereich des Inhalts markiert (per Maus oder Tastatur), so wird die Markierung (nicht der markierte Inhalt!) an alle Teilnehmer übertragen und angezeigt – sofern der zur Markierung gehörende Dokumentenbereich dort ganz oder teilweise sichtbar ist.

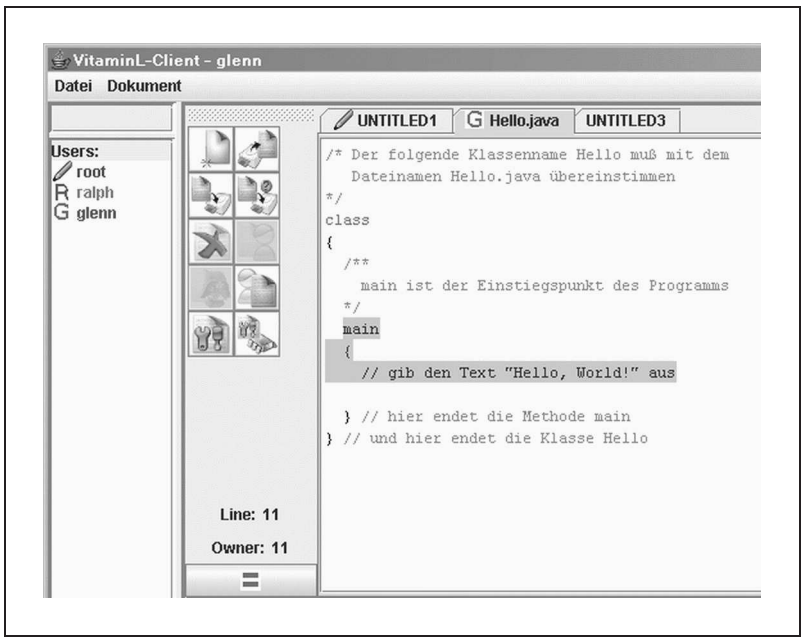

Abb. 8.14: Informationsfunktionen in der VitaminL-IDE
(Screenshot)

Zur besonderen Unterstützung der Java-Programmierung wurden in den Gruppeneditor weitere spezifische Funktionalitäten integriert:

[2] Die Ausgabe von Zeilennummer vor jeder Textzeile eines Dokuments konnte erst in einer späteren Version der VitaminL-IDE umgesetzt werden.

- *Syntax Highlighting*:
  Die besondere Hervorhebung von Elementen der Programmiersprache Java in Abhängigkeit von ihrer Bedeutung in unterschiedlichen Farben und Schriftstilen verbessert im Allgemeinen die Lesbarkeit von Quelltexten.

- Java-Compiler:
  Die bereits im Prototyp vorhandene Integration des Java-Compilers wurde verbessert, so dass unter anderem eine Geschwindigkeitserhöhung beim Übersetzen von Quelltexten erreicht werden konnte.

- Java-Interpreter:
  Eine einfache Variante einer Shell in Verbindung mit der Möglichkeit, den Java-Interpreter aus der VitaminL-IDE heraus zu starten, vereinfacht die Ausführung eigener, in VitaminL entwickelter Java-Klassen.

8.3.2 Ergebnisse

Der neugestaltete Gruppeneditor kam erstmals im Dezember 2004 zum Einsatz, zeitgleich mit der überarbeiteten Dialogschnittstelle zur strukturierten Kommunikation (s. Kap.8.2.4). Die von den Teilnehmern erreichten Arbeitsergebnisse variieren hinsichtlich des Erfüllungsgrads der gestellten Aufgabe, entsprechen aber unter Berücksichtigung der für alle Probanden ungewohnten Lernsituation – es ist zu bedenken, dass alle Teilnehmer erstmalig unter Verwendung der VitaminL-IDE zusammenarbeiten – und des recht knapp bemessenen Zeitrahmens von 90 Minuten für die eigentliche Aufgabenbearbeitung (s. Tab.8.1) durchaus den Erwartungen. Dies allein mag schon als Indiz dafür gelten, dass ein erfolgreiches Zusammenarbeiten mittels der VitaminL-IDE und insbesondere dem überarbeiteten Gruppeneditor eine realistische Annahme darstellt.

Analysiert man darüber hinaus die eingegangenen Fragebögen der Teilnehmer im Hinblick auf den Gruppeneditor, so sind die Rückmeldungen als durchaus positiv zu bewerten. So wird besonders *„die Möglichkeit der Einsicht der von anderen Teammitgliedern erstellten Dokumente"* betont, verbunden mit dem *„schnellen und unkomplizierten Tauschen, Bearbeiten und Zurverfügungstellen von Dokumenten"*. Auch der Aspekt der Ortsunabhängigkeit wird erkannt und von den Teilnehmern begrüßt, da *„man mit Hilfe dieses Programms Distanzen bei de Entwicklung eines Projekts überwinden kann. Außerdem muss man sich nicht um einen einzelnen PC drängeln, um etwas zu dem Projekt beizutragen."*

Insgesamt enthalten die 21 eingegangenen Fragebögen zwölf positive und zwei neutrale Aussagen hinsichtlich der Unterstützung der Zusammenarbeit durch den Gruppeneditor, negative Kritik ist nicht vorhanden. Der Gruppeneditor von VitaminL wird in der nun vorliegenden Version von den Teilnehmern akzeptiert und ermöglicht ein angemessenes Zusammenarbeiten bei der Java-Programmierung, wovon auch der zugehörige Lernprozess profitiert.

8.4 Architektur des VitaminL-CSCL-Systems

Basierend auf den Ergebnissen der durchgeführten Studien und den daraus gewonnenen Erkenntnissen wurde im VitaminL-Projekt eine CSCL-Applikation entwickelt, die im Sommersemester 2005 im Rahmen einer Kooperationsveranstaltung zum Thema *„Programmierung in Java"* zwischen den Hochschulen Konstanz und Hildesheim produktiv zum Einsatz kam.

8.4.1 Client-Server-Ansatz

Die VitaminL-Applikation arbeitet nach dem klassischen Client-Server-Modell: *„Im sogenannten Client/Server-Modell erhält ein Clientprogramm, das auf einem Endsystem läuft, Informationen von einem Server, der auf einem anderen Endsystem läuft"* (Kurose & Ross, 2002, S.28). Die Verbindung zwischen den beteiligten Rechnern erfolgt über das Internet.

8.4.1.1 Der VitaminL-Server

Der VitaminL-Server *„ist ein dediziertes Programm mit dem speziellen Zweck, einen Dienst bereitzustellen, kann aber gleichzeitig mehrere entfernte Clients bedienen"* (Comer, 2002, S.406). Vollständig in Java 5.0 implementiert ist der VitaminL-Server plattformunabhängig und kann praktisch auf jedem Rechner betrieben werden, der eine passende Laufzeitumgebung (JRE V1.5 oder höher) zur Verfügung stellt. Das Produktivsystem läuft derzeit auf einem Linux-Rechner (unter SuSE Linux 9.2), die erwähnte Plattformunabhängigkeit erlaubt aber auch eine Ausführung auf einem Arbeitsplatzrechner mit einem *MS Windows*-Betriebssystem, was sich während der Entwicklungsphase zu Testzwecken als vorteilhaft erwiesen hat: Neue Konzepte und Fehlerkorrekturen können lokal getestet werden, ohne den laufenden Betrieb in irgendeiner Weise negativ zu beeinflussen.

Da die Server-Software von VitaminL, sobald sie einmal auf einem Host gestartet ist und ausgeführt wird, entsprechend der für Server-Anwendungen typischen Arbeitsweise *„passiv auf die Verbindungsaufnahme durch einen entfernten Client"*(Comer, 2002, S.406) wartet und dabei ohne jegliche Benutzereingriffe abläuft (mit Ausnahme des Stops oder Neustart der Server-Anwendung bspw. nach einer Aktualisierung der Software), ist der VitaminL-Server als Konsolenapplikation realisiert, die – im Gegensatz zur VitaminL-IDE – keine Benutzungsoberfläche besitzt. Stattdessen werden wichtige Ereignisse als Text in der Standardausgabe angezeigt (s. Abb.8.15) oder in Protokolldateien auf der Festplatte des Hosts geschrieben.

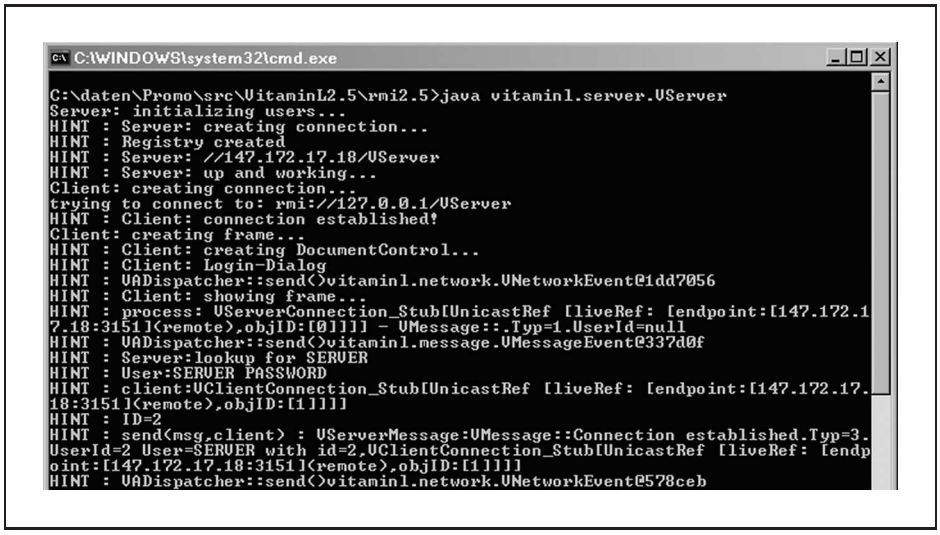

Abb. 8.15: Die Server-Applikation von VitaminL
(Screenshot)

Als zentrale Komponente der VitaminL-Software ist die Server-Anwendung für folgende (Teil-)Aufgaben – und damit die Bereitstellung zugehöriger Dienste – zuständig:

- Verwaltung von Benutzern und Benutzergruppen,
- Verwaltung von Sitzungen,
- Protokolldateien,
- Dokumenten-Management und
- Kommunikation.

Die Umsetzung des Servers und seiner Dienste innerhalb der VitaminL-Software soll zum besseren Verständnis für spätere Kapitel nachfolgend kurz skizziert werden.

8.4.1.1.1 Verwaltung von Benutzern und Gruppen Für das Arbeiten mit VitaminL ist eine Kennung – bestehend aus Benutzername und Passwort – notwendig. Mit solch einer Kennung authentifiziert sich jeder Anwender nach dem Start seiner Client-Applikation gegenüber dem System, in dem er diese Kennung dem Server übermittelt und sich dort anmeldet. Die zur Überprüfung einer Kennung notwendige Information liest der Server[3] beim Start aus einer Datenbank aus, welche neben dem für eine Authentifizierung notwendigen Passwort weitere Informationen wie beispielsweise die Gruppe eines jeden Benutzers, das ihm zugeordnete Icon oder auch die Textfarbe des Benutzers (zur Unterscheidung von empfangenen

[3] Wenn nicht anders angegeben, wird der Begriff *Server* (oder auch *VitaminL-Server*) synonym für die Server-Applikation der VitaminL-Software verwendet.

Textnachrichten innerhalb der während einer Sitzung stattfindenden Kommunikation). Bestimmte Benutzer mit speziellen Rechten (bspw. Tutoren) sind – im Unterschied zu allen anderen Benutzern – keiner Gruppe fest zugeordnet, sondern können diese nach Belieben im laufenden Betrieb wechseln. Auf diese Weise kann ein (menschlicher) Tutor bei Bedarf mehrere Gruppen gleichzeitig betreuen.

8.4.1.1.2 Verwaltung von Sitzungen Meldet sich ein Teilnehmer erfolgreich am Server an, so wird automatisch für die Gruppe, der er zugeordnet wurde oder – im Falle von Tutoren – beigetreten ist, eine neue Sitzung gestartet, sofern nicht bereits ein anderer Benutzer mit dieser Gruppe am System angemeldet ist. Mit dem Start einer Sitzung werden – für die zugehörige Gruppe – verschiedene Komponenten initialisiert:

- Eine **Protokolldatei** wird erzeugt, in welcher sämtliche Aktionen der Gruppenmitglieder während der Sitzung dauerhaft gespeichert werden. Diese Protokolldateien können nach Ablauf einer Sitzung verwendet werden, beispielsweise um die Sitzung – ähnlich einer Art Videoaufzeichnung – wiederholt ablaufen zu lassen und dabei bestimmte Aspekte der Zusammenarbeit zu beobachten oder aber um die Aktivitäten der Teilnehmer zu analysieren.
- Ein **Dokumenten-Manager** dient der Verwaltung aller (Quelltext-)Dokumente, die von den Gruppenmitgliedern zur Bearbeitung erzeugt beziehungsweise geöffnet wurden. Diese Komponente informiert alle angemeldeten Clients (einer Gruppe) über Änderungen von Dokumenteninhalten und -namen. Sie regelt weiterhin die Freigaben, Anforderungen und sonstigen Operationen der Gruppenmitglieder, die die gemeinsamen Dokumente betreffen, gemäß dem in Kapitel 8.3 skizzierten Bedienkonzept.
- Eine **Kommunikationsverwaltung** organisiert sämtliche Textnachrichten, die während einer Sitzung zwischen den Gruppenmitgliedern (mittels der strukturierten Dialogschnittstelle; s. Kap.8.2) ausgetauscht werden. Insbesondere werden nach einer Anmeldung alle bislang in der Sitzung ausgetauschten Nachrichten an das neue Gruppenmitglied geschickt, so dass auch Mitglieder, die einer bereits längere Zeit aktiven Sitzung beitreten, über die gesamte bis dato stattgefundene Kommunikation informiert werden.

Sobald sich der letzte Teilnehmer einer Sitzung vom Server abmeldet, wird die Sitzung beendet, die zugehörige Protokolldatei wird geschlossen und alle anderen zu der Sitzung gehörenden Komponenten werden wieder aus dem System entfernt.

8.4.1.2 Der VitaminL-Client

Die Client-Applikation von VitaminL[4] stellt dem Anwender die Benutzungsoberfläche zum gemeinsamen Lernen und Arbeiten mit VitaminL zur Verfügung.

[4] Nachfolgend auch *Client* oder *VitaminL-Client* genannt.

Wie der Server ist auch der Client vollständig in Java entwickelt, besitzt aber – im Gegensatz zum Server – eine GUI, die unter Verwendung der Swing-Klassen geschrieben wurde. Somit soll dem Anwender eine unter software-ergonomischen Aspekten angemessene und zeitgemäße Bedienung des Programms ermöglicht werden.

Da wesentliche Bedienkonzepte bereits vorgestellt wurden (s. Kap.8.2: Strukturierte Dialogschnittstelle; s. Kap.8.3: Gruppeneditor), sei an dieser Stelle eine kurze Zusammenfassung des Clients und wichtiger Elemente erlaubt.

8.4.1.2.1 Anmeldung am Server Nach dem Start des Clients führt der Anwender üblicherweise ein Login durch, das heißt er meldet sich mit seiner Kennung an einem laufenden VitaminL-Server an (s. Abb.8.16(a)). Anwender mit Tutor-Status können in einem anschließenden Dialog die Gruppe, der sie beitreten möchten, auswählen (s. Abb.8.16(b): Bereits aktive Gruppen sind farbig hervorgehoben und am Beginn der alphabetisch sortierten Liste positioniert), alle anderen Anwender sind einer Gruppe fest zugeordnet.

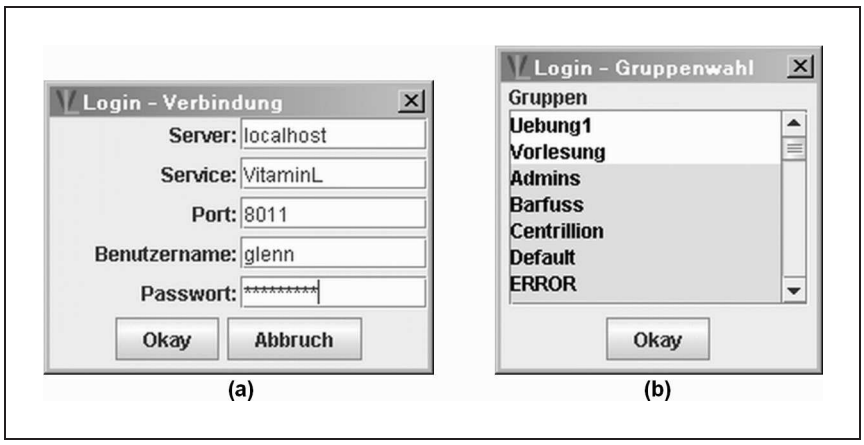

Abb. 8.16: Anmeldung eines Clients an einem VitaminL-Server (Screenshot)

8.4.1.2.2 Gruppenkommunikation Die Kommunikation innerhalb einer Gruppe ist als Chat mit der in Kapitel 8.2.4 vorgestellten Dialogschnittstelle realisiert. Da stets alle Nachrichten an das gesamte Team versendet werden – auch bei Auswahl eines dedizierten Empfängers – , herrscht hinsichtlich der Kommunikation eine totale Offenheit, wodurch private Absprachen und Geheimnisse verhindert werden. Alle Teammitglieder sind im Sinne von echter Teamarbeit gleichermaßen in die Gruppenkommunikation einbezogen und somit während einer Sitzung umfassend über die Planung und den Verlauf informiert.

8.4.1.2.3 Kooperationsunterstützung Die Kooperation eines virtuellen Teams erfolg im Gruppeneditor, der zentral für eine Gruppe auf dem Server verwaltet und auf allen Clients der Gruppe gespiegelt wird. Infolgedessen werden

sämtliche Dokumentenoperationen eines Clients zunächst an den Server geschickt und dort verarbeitet; das Ergebnis wird anschließend vom Server an alle Clients gesendet und dort weiter verarbeitet. Auf diese Weise werden sämtliche Dokumente einer Sitzung durch den Server synchronisiert.

8.4.1.2.4 Awareness-Funktion Alle aktiven Mitglieder der Gruppe werden mitsamt den ihnen zugeordneten Symbolen in einer Liste am linken unteren Rand der Client-Applikation angezeigt (s. Abb.8.17). Damit wird einerseits die für eine erfolgreiche Teamsitzung notwendige Awareness-Funktion erfüllt und andererseits eine Hilfe bei der Zuordnung von Dokumenten im Gruppeneditor zu Benutzern gegeben (s. Kap.8.3).

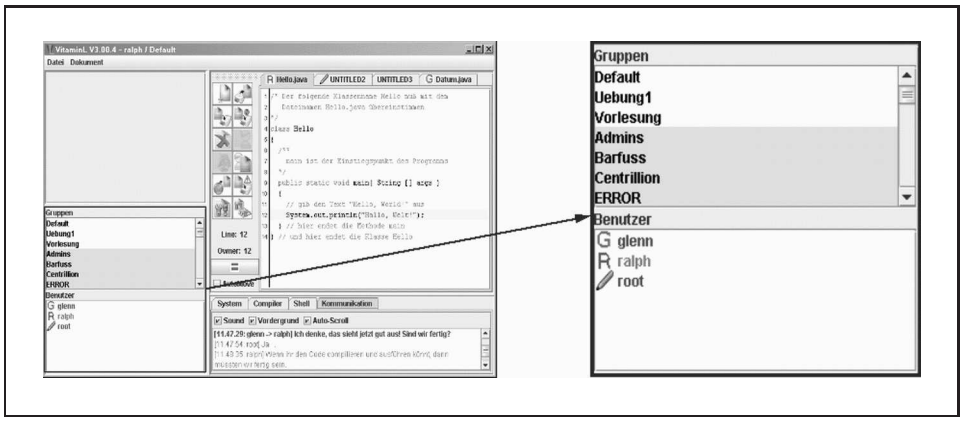

Abb. 8.17: Benutzer- und Gruppeninformationen im VitaminL-Client (Screenshot)

8.4.1.2.5 Zusätzliche Informationsbereiche Am unteren Rand des Clients sind mehrere Informationsfenster positioniert, die sich einen gemeinsamen Bereich des Bildschirms teilen und dem Anwender diverse sitzungsspezifische Informationen (in Form von textbasierten Ausgaben) offerieren.

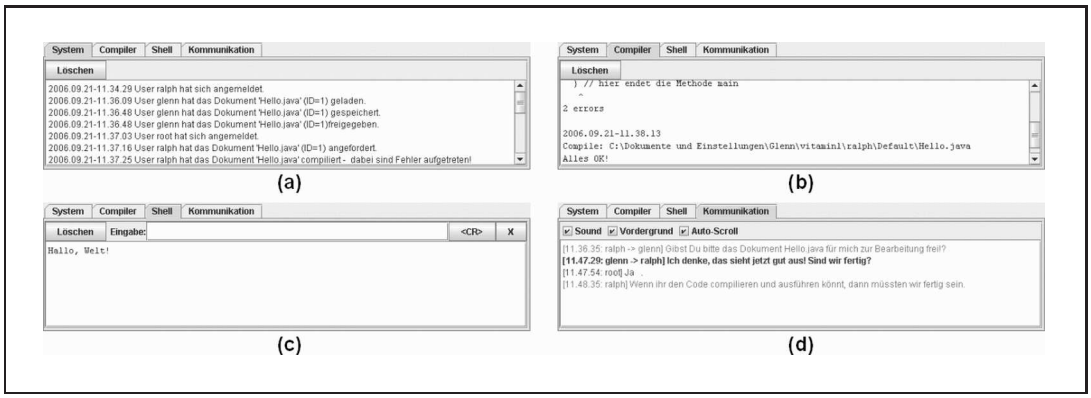

Abb. 8.18: Zusätzliche Informationsbereiche im VitaminL-Client (Screenshot)

Zu diesem Zwecke sind dort derzeit vier Fenster mit den Titeln *System*, *Compile*, *Shell* und *Kommunikation* definiert. Über registerähnliche Bedienelemente kann je eines dieser Fenster ausgewählt, in den Vordergrund geholt und angezeigt werden.

Das Informationsfenster *System* Verschiedene Nachrichten des VitaminL-Systems wie beispielsweise An- und Abmeldungen anderer Teammitglieder oder auch die Ergebnisse von Compiler-Aufrufen der Teammitglieder (inklusive möglicher Fehlermeldungen) werden in diesem Fenster (s. Abb.8.18(a)) angezeigt.

Das Informationsfenster *Compile* In diesem Teilfenster (s. Abb.8.18(b)) werden die Meldungen des Compilers ausgegeben.

Das Informationsfenster *Shell* Im Zuge der Integration des Java-Interpreters in die VitaminL-IDE wird in diesem Fenster (s. Abb.8.18(c)) eine einfache Shell bereitgestellt. Dort werden einerseits die Textausgaben von Java-Applikationen angezeigt, die aus der VitaminL-IDE heraus gestartet worden sind, und andererseits Texteingaben des Benutzers (über ein zusätzliches Eingabefeld) an die laufende Applikation übergeben. Ferner existiert die Möglichkeit, das laufende Java-Programm über die Schaltfläche *X* abzubrechen – dies ist insbesondere dann sinnvoll und hilfreich, wenn das Programm aufgrund fehlerhafter Programmierung nicht korrekt zur Terminierung gelangen kann.

Das Informationsfenster *Kommunikation* Das Kommunikationsfenster (s. Abb.8.18(d)) dient der Ausgabe sämtlicher eingehender Nachrichten, die zwischen den Teammitgliedern zum Zwecke der Kommunikation ausgetauscht werden. Zusätzliche Bedienelemente ermöglichen weitere Einstellungen:

- Option *Sound*:
  Über diesen Schalter kann das akustische Signal ein- beziehungsweise ausgeschaltet werden, das – wenn es aktiviert ist – immer dann ertönt, wenn eine eingehende Nachricht den aktuellen Anwender als dedizierten Empfänger aufweist.
- Option *Vordergrund*:
  Wenn diese Option gewählt ist, wird das Kommunikationsfenster bei Empfang einer neuen Nachricht automatisch in den Vordergrund gebracht.
- Option *Auto-Scroll*:
  Bei eingeschaltetem *Auto-Scroll* wird der Fensterausschnitt bei eingehenden Nachrichten stets auf das Textende gesetzt, so dass die letzte Nachricht sofort angezeigt wird und vom Empfänger gelesen werden kann.

Da allen Mitgliedern eines Teams unterschiedliche Farben zugeordnet sein können, die auch für die Darstellung von Kommunikationsbeiträgen verwendet werden, ist eine leichtere Zuordnung dieser Beiträge zu ihren Autoren möglich.

8.4.2 Nachrichtenbasierte Client-Server-Kommunikation

Die Kommunikation zwischen Server- und Client-Komponenten erfolgt mittels spezieller Nachrichtenobjekte, die über socket-basierte Netzwerkverbindungen zwischen den Kommunikationspartnern (Server und Clients) ausgetauscht werden. Unter einem Socket wird allgemeinhin ein Endpunkt einer Kommunikationsverbindung zwischen zwei Rechnern verstanden (vgl. Boger, 1999, S.47). Der Aufbau einer Netzwerkverbindung zwischen Client und Server wird vom Client während eines Login (s. Kap.8.4.1.2.1) initiiert, die nachfolgende Kommunikation wird nach dem Client/Server-Paradigma (vgl. Comer, 2002, Kap.3.3f) beziehungsweise dem Client/Server-Modell abgewickelt: *„Die Kommunikation findet in der Weise statt, dass der Client-Prozess über das Netzwerk eine Nachricht an den Server-Prozess schickt. Der Client-Prozess wartet dann auf die Antwort. ... Wenn der Server die Anforderung erhält, führt er die angeforderte Aufgabe aus oder ermittelt die angeforderten Daten und sendet eine Antwort zurück"* (Tanenbaum, 2003a, S.18). Die Abb.8.19 stellt dieses Kommunikationsmodell vereinfacht dar. Dieser traditionelle Ansatz wurde in der VitaminL-Software dahingehend erweitert, dass auch der Server tätig werden kann und von sich aus Nachrichten an Clients schickt, die sich allerdings zuvor bei ihm angemeldet haben müssen.

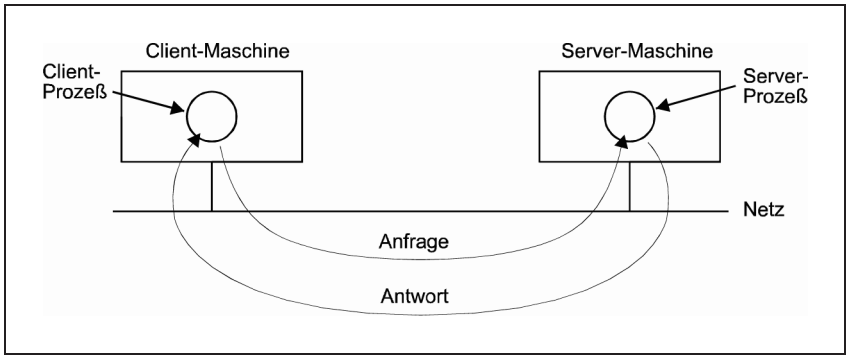

Abb. 8.19: Das Client/Server-Modell
(Tanenbaum, 2003a, S.19)

Der innere Aufbau der an diesem Kommunikationsparadigma beteiligten Programme (d.h. Server- oder Client-Applikation) lässt sich in Form eines Schichtenmodells beschreiben:

1. Die unterste Schicht ist die **Netzwerkschicht**. Hier wird der eigentliche Austausch von Nachrichtenobjekten zwischen den Kommunikationspartnern durchgeführt, das heißt die Netzwerkschicht nimmt zum einen Objekte aus der darüberliegenden Schicht (der Steuerungsschicht; s. Punkt 2) entgegen und verschickt sie über eine Netzwerkverbindung an die Gegenstelle der Kommunikation und leitet andererseits über das Netzwerk empfangene Nachrichtenobjekte an die darüberliegende Schicht weiter.

2. Oberhalb der Netzwerkschicht ist die **Steuerungsschicht** angesiedelt. In dieser Schicht ist die prinzipielle Arbeitsweise der jeweiligen Applikation (Server bzw. Client) realisiert. Eingehende Nachrichten werden analysiert und an

zugehörige Komponenten in höherliegende Schichten weitergereicht. Ergebnisse aus diesen oberen Schichten werden in Nachrichtenobjekte verpackt und der Netzwerkschicht zum Versand an relevante Kommunikationspartner übergeben.

3. In der über der Steuerungsschicht liegenden **Komponentenschicht** sind wesentliche Komponenten wie beispielsweise die Dokumentenorganisation oder die Kommunikationsverwaltung angesiedelt, die jeweils spezielle Aspekte der Applikation realisieren. Eingehende Nachrichten werden hier verarbeitet, Ergebnisse der Verarbeitung werden an die tieferliegende Steuerungsschicht zurückgeliefert oder an die oberste Schicht (die GUI-Schicht) zur Anzeige weitergereicht.

4. Die **GUI-Schicht** in Form einer graphischen Oberfläche ist nur in der Client-Applikation vorhanden. Ihre Aufgabe besteht zunächst darin, Zustände der tiefergelegenen Komponenten in geeigneten GUI-Elementen darzustellen. Darüber hinaus stellt sie dem Anwender Bedienelemente (Schaltflächen, Menüs etc.) zur Verfügung, über die er Befehle auslösen kann, die an die zugehörige Komponente weitergereicht werden und dort in geeignete Aktionen umgewandelt und weiterverarbeitet werden. Dies hat üblicherweise die Erzeugung und Weiterleitung von Nachrichtenobjekten über tiefergelegene Schichten zur Folge.

Die nachfolgende Abbildung stellt dieses Schichtenmodell graphisch dar. Die eingezeichneten Pfeile sollen aufzeigen, in welchen Richtungen zwischen den diversen Schichten kommuniziert wird.

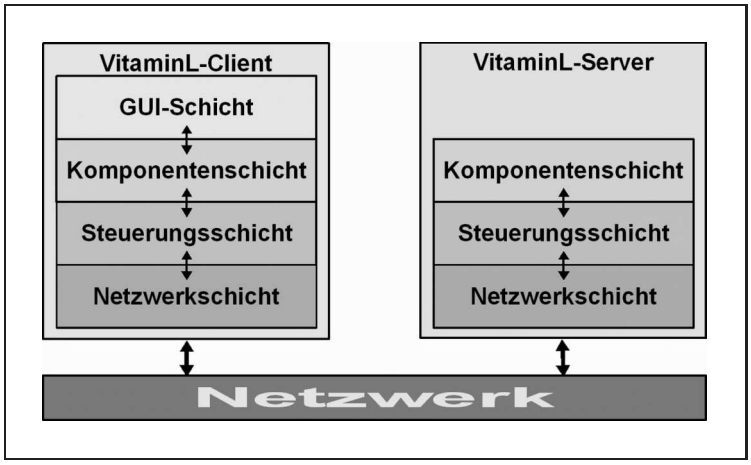

Abb. 8.20: Das Schichten-Modell der VitaminL-Applikationen

Die Steuerungsschicht einer VitaminL-Applikation repräsentiert die zentrale Ablaufsteuerung und entscheidet darüber, wie eingehende Nachrichten weiterverarbeitet werden. Die interne Arbeitsweise dieser Schicht, sowohl auf dem Server als auch auf den Clients, lässt sich – stark vereinfacht – durch Deterministische Endliche Automaten (DEAs; vgl. Kastens & Kleine Büning, 2005, Kap.7.2; Hopcroft et al., 2002, Kap.2; Rechenberg, 1999, Kap.A3) beschreiben:

- Jede Komponente (Server wie Client) befindet sich stets in einem wohldefinierten Zustand q aus einer endlichen Zustandsmenge Q.

- Eine empfangene Nachricht e stellt im Sinne der Automatentheorie ein Zeichen aus dem Eingabealphabet Σ dar und löst eine Transition aus, das heißt der Automat wechselt in einen Folgezustand.

- Mit der Transition kann eine weitere Aktion a (oder eine Folge von Aktionen) ausgelöst, die in der Automatentheorie als Ausgabezeichen eines endlichen Ausgabealphabets Δ definiert wird.

Die Nachrichtenobjekte nehmen nicht nur in diesem Modell eine zentrale Stellung ein, sondern sind auch von größter Bedeutung für die späteren Analysekonzepte zur Erkennung von Teamrollen einerseits und Problemsituationen andererseits.

Die zwischen Client und Server ausgetauschten Nachrichtenobjekte modellieren die Anforderungen und Antworten des Client/Server-Modells (s. Abb.8.19) und werden als Instanzen von speziellen Java-Klassen erzeugt, deren Aufbau, hierarchische Beziehungen und Arbeitsweise nachfolgend kurz skizziert werden. Damit soll eine Wissensgrundlage für ein besseres Verständnis nachfolgender Kapitel geschaffen werden.

8.4.2.1 Die Basisklasse `VMessage`

Die Klasse `VMessage` stellt als sogenannte Basis- oder Oberklasse einer Klassenhierarchie den Prototypen sämtlicher Client/Server-Nachrichten dar, das heißt die Definition der Klasse `VMessage` legt Attribute (Eigenschaften) und Operationen (Verhaltensweisen) fest, die für sämtliche Nachrichtenobjekte, die auf Basis dieser Klasse erzeugt werden, Gültigkeit besitzen.

8.4.2.1.1 Attribute der Klasse `VMessage` Jedes Nachrichtenobjekt der VitaminL-Software besitzt eine Menge von Eigenschaften, die durch die folgenden Attribute beschrieben werden:

- `String timestamp`:
  Der Zeitstempel einer Nachricht gibt an, wann sie erzeugt und verschickt wurde.

- `VUserID senderId`:
  Der Absender einer Nachricht wird mittels einer eindeutigen Benutzerkennung in jeder Nachricht notiert.

- `int type`:
  Jede Nachricht einer Nachrichtenklasse kann anhand eines Nachrichtentyps näher spezifiziert werden.

- `int code`:
  Jeder Nachricht ist ferner ein Code zugeordnet, der aus einer Erweiterung des Ansatzes der CLS resultiert.

Zwar besitzt die Klasse `VMessage` weitere, zum Teil von ihrer Oberklasse `Object` übernommene Attribute, diese sind aber für das Verständnis der vorliegenden Arbeit nicht relevant und werden daher an dieser Stelle nicht weiter aufgeführt.

8.4.2.1.2 Operationen der Klasse `VMessage` Die Operationen einer Klasse legen die Verhaltensweisen der zugehörigen Objekte fest und werden in Java mit Methoden (bestehend aus Methodenname, Parameterliste, Rückgabetyp und Anweisungen) umgesetzt. Neben Konstruktoren für die Objekterzeugung und Zugriffsmethoden für den Schreib- beziehungsweise Lesezugriff auf Attribute einer Nachricht sind in der Nachrichtenklasse `VMessage` unter anderem folgende Operationen definiert:

- `void process(VIProcessor proc)`:
  Mit dieser Methode verarbeitet sich eine Nachricht selber. Dazu ruft sie – je nach Typ der Nachricht – entsprechende Methoden eines sogenannten *Prozessors* auf, der in Form einer Schnittstelle vom Typ `VIProcessor` (oder Ableitungen) spezifiziert und als Parameter angegeben ist. Diese Methode ist in Ableitungen von `VMessage` geeignet zu überschreiben.
- `String toXMLString()`:
  Der Aufruf dieser Methode liefert eine Zeichenkette zurück, die die aktuelle Nachricht in einem XML-Format darstellt. Diese Methode muss für den Protokoll-Mechanismus in jeder Ableitung von `VMessage` geeignet überschrieben werden. Dieser spezielle Aspekt wird in Kapitel 8.4.4.2 ausführlicher behandelt werden.

8.4.2.1.3 Schnittstellen der Klasse `VMessage` Schnittstellen oder Interface-Klassen enthalten in der Regel abstrakte Operationen in Form von Methodendeklarationen, die in implementierenden Klassen vollständig mit dem jeweils gewünschten Verhalten umzusetzen sind. In der Klasse `VMessage` sind die Schnittstellen `Serializable` und `VILoggable` implementiert.

Die Schnittstellenklasse `Serializable` Damit Nachrichtenobjekte über eine Netzwerkverbindung geschickt werden können, müssen sie serialisierbar sein. Diese Eigenschaft wird durch Implementierung der (leeren) Schnittstelle `Serializable` für die Klasse `VMessage` und ihre Ableitungen sichergestellt.

Die Schnittstellenklasse `VILoggable` Diese Schnittstelle dient der Spezifikation von Objekten, die den Protokoll-Mechanismus der VitaminL-Software direkt unterstützen, indem sie die in dieser Schnittstelle deklarierte Operation `String`

`getLogString()` umsetzen und damit zur Verfügung stellen. Diese Operation liefert einen geeigneten, detailierten String zurück, der im Anschluss in eine Protokolldatei geschrieben werden kann. Intern greift diese Operation auf die Methode `String toXMLString()` zu, die in allen Nachrichtenklassen spezifisch umgesetzt ist.

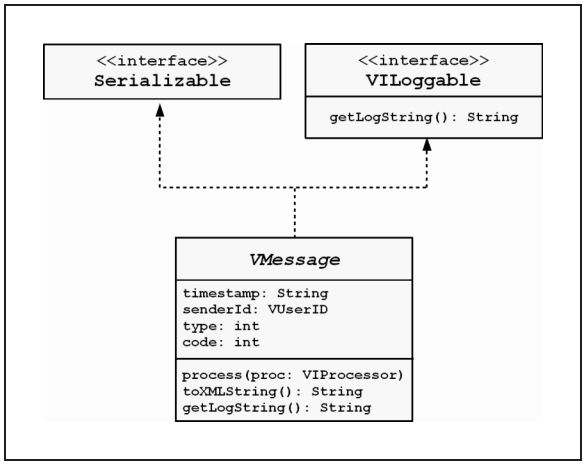

Abb. 8.21: UML-Klassendiagramm der Klasse `VMessage` (Auszug)

8.4.2.2 Abgeleitete Nachrichtenklassen

Spezielle Aspekte der Client-Server-Kommunikation werden mit Ableitungen der Nachrichtenklasse `VMessage` umgesetzt. Gemäß dem objektorientierten Konzept der Vererbung dient die Basisklasse als eine Art Vorlage und reicht zunächst sämtliche Attribute und Operationen an alle Unterklasse weiter. In jeder dieser Unterklassen können nun weitere Attribute und/oder Operationen hinzugefügt werden[5], des weiteren können geerbte Operationen in den Unterklassen re-definiert werden: Das sogenannte *Überschreiben von Methoden* ermöglicht es, das in einer Oberklasse vordefinierte Verhalten für Objekte von Unterklassen anzupassen beziehungsweise völlig neu zu gestalten (vgl. Balzert, 1999, S.336; Schiedermeier, 2005, S.247ff).

Insgesamt betrachtet stellen die Unterklassen und deren Objekte Spezialfälle der gemeinsamen Oberklasse (hier: `VMessage`) dar.

8.4.2.2.1 Die Ableitung `VCommunicationMessage` Diese Unterklasse von `VMessage` kann als eine Art Container aufgefasst werden für sämtliche Nachrichten, dier der (textbasierten) Kommunikation zwischen den Teilnehmern einer VitaminL-Sitzung dienen.

Zusätzlich zu den Attributen ihrer Oberklasse besitzt diese Klasse weitere Attribute:

[5] Die Unterklasse bildet dann eine echte Erweiterung der Oberklasse

- `VUserID receiverId`:
  Dieses Attribut enthält die Kennung eines dedizierten Empfängers, sofern vor dem Versenden dieser Botschaft ein Empfänger ausgewählt wurde.

- `Object content`:
  Der eigentliche (Text-)Inhalt einer Botschaft wird mit diesem Attribut modelliert.

8.4.2.2.2 Die Ableitung `VDocumentMessage` Die Klasse `VDocumentMessage` als Ableitung der Klasse `VMessage` ist ihrerseits Oberklasse für eine Menge von speziellen Klassen, die in der VitaminL-Software im Rahmen des Nachrichtenaustauschs zwischen Client und Server zur Verwaltung und Bearbeitung von Textdokumenten innerhalb des Gruppeneditors erzeugt und verschickt werden. Dazu erweitert die Klasse `VDocumentMessage` ihre Oberklasse um das Attribut `VDocumentID documentID`, welches – analog zu Benutzerkennungen – zur eindeutigen Identifikation von Dokumenten innerhalb einer Gruppensitzung dient. Mehrere Ableitungen erweitern diese Klasse, um somit spezielle Aspekte des *Shared Documents*-Konzept von VitaminL zu realisieren.

8.4.2.2.3 Hierarchie der Nachrichtenklassen Neben den soeben skizzierten Klassen für Kommunikations- und Dokumentennachrichten existieren weitere Ableitungen der Klasse `VMessage` beispielsweise für Systemnachrichten (mit Informationen über An-/Abmeldevorgänge von Gruppenmitgliedern), für Servernachrichten (zum Überprüfen der Gültigkeit von Verbindungen zu angemeldeten Clients oder auch zur Aktualisierung von Benutzer- und Gruppenlisten) oder auch für die Ausführung spezieller administrativer Tätigkeiten (wie eine Aktualisierung der Datenbank). Die resultierende Klassenhierarchie ist in vereinfachter Darstellung der Abbildung 8.22 zu entnehmen.

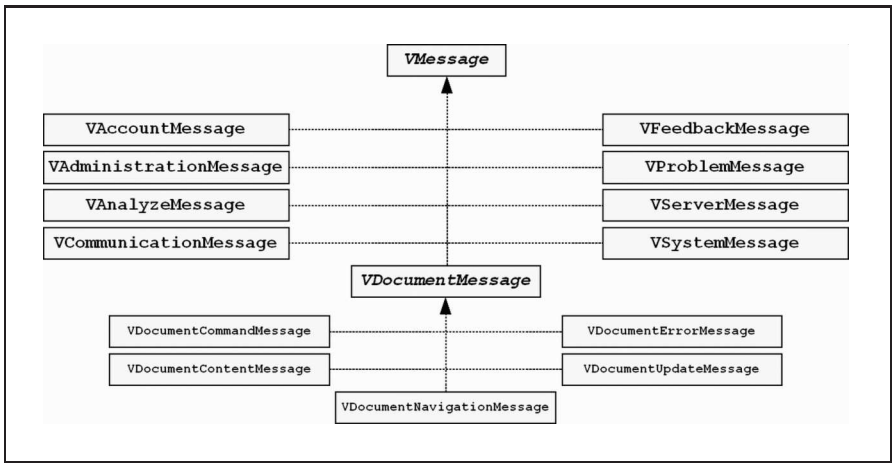

Abb. 8.22: UML-Klassendiagramm der VitaminL-Nachrichtenklassen

8.4.2.3 Weitere Klassen der Client-Server-Kommunikation

Neben den genannten Nachrichtenklassen sind in der VitaminL-Software weitere Klassen definiert, die für die nachrichtenbasierte Client-Server-Kommunikation relevant sind. Zum einen handelt es sich dabei um eine Hierarchie von Schnittstellenklassen, mit denen die Operationen für die Verarbeitung der einzelnen Nachrichtenklassen spezifiziert werden, zum anderen gehören dazu auch die Warteschlangen, die den Versand und Empfang von Nachrichten unterstützen.

8.4.2.3.1 Verarbeitung von Nachrichtenobjekten Unterschiedliche Nachrichten unterscheiden sich auch in der Art und Weise ihrer Verarbeitung. Dabei weiß im Prinzip jede Nachrichtenklasse selbst am besten, wie ihre Objekte zu verarbeiten sind. So wurde analog zu der Hierarchie der Nachrichtenklassen eine Hierarchie von Schnittstellenklassen definiert, mit welcher für jede Nachrichtenklasse geeignete Operationen für die Verarbeitung ihrer Objekte spezifiziert werden.

Die Schnittstelle `VIProcessor` Die Schnittstelle `VIProcessor` legt fest, wie Nachrichtenobjekte vom Typ `VMessage` zu verarbeiten sind. Da die Klasse `VMessage` abstrakt ist, können von ihr keine Objekte erzeugt und damit auch nicht verarbeitet werden. Infolgedessen enthält die Schnittstelle `VIProcessor`, die den zugehörigen Verarbeitungsmechanismus spezifiziert, zunächst keine Operationen, sie ist folglich leer. Sie ist jedoch – analog zur Nachrichtenklasse `VMessage` – die Oberklasse einer Hierarchie von Nachrichtenverarbeitern.

Ableitungen der Schnittstelle `VIProcessor` Sämtliche Ableitungen von `VIProcessor` stellen Spezialfälle dieser Schnittstelle dar und spezifizieren somit die Operationen für entsprechende Nachrichtenklassen als Ableitungen von `VMessage`.

Tab. 8.3: Zuordnung von Nachrichtenklassen zu Verarbeitungsschnittstellen

Nachrichtenklasse	Verarbeitungsklasse
`VMessage`	`VIProcessor`
`VAccountMessage`	`VIAccountProcessorServer`
`VAdministrationMessage`	`VIAdministrationMessageProcessorServer`
`VAnalyzeMessage`	`VIAnalyzeProcessor`
`VCommunicationMessage`	`VICommunicationProcessor`
`VDocumentCommandMessage`	`VIDocumentCommandProcessor`
`VDocumentContentMessage`	`VIDocumentContentProcessor`
`VDocumentErrorMessage`	`VIDocumentErrorProcessor`
`VDocumentNavigationMessage`	`VIDocumentNavigationProcessor`
`VDocumentUpdateMessage`	`VIDocumentUpdateProcessor`
`VFeedbackMessage`	`VIFeedbackProcessor`

Fortsetzung auf nächster Seite

Tab. 8.3: Zuordnung von Nachrichtenklassen zu Verarbeitungsschnittstellen *Forts.*

Nachrichtenklasse	**Verarbeitungsklasse**
`VProblemMessage`	`VIProblemProcessor`
`VServerMessage`	`VIServerProcessorClient`
`VSystemMessage`	`VISystemProcessorClient`

Sämtliche Schnittstellen für die Nachrichtenverarbeitung sind direkt von `VIProcessor` abgeleitet. Anhand des Postfixs im Namen (`...Server` oder `...Client`) ist erkennbar, ob eine Nachrichtenverarbeitung exklusiv auf dem Server oder dem Client erfolgt (ohne entsprechendes Postfix kann die zugehörige Nachricht auf beiden Applikationen verarbeitet werden).

Beispiel 8.1: Quellcode der Schnittstelle `VIDocumentCommandProcessor`

```
interface VIDocumentCommandProcessor extends VIProcessor
{
  /**
   * Schliessen eines Dokuments
   *
   * @param senderid    der Absender der Botschaft
   * @param documentid  die ID des zu schliessenden Dokuments
   */
  void msgCommandDocumentClose(VUserID senderid, VDocumentID documentid);

  /**
   * Uebersetzen eines Dokuments per Compiler
   *
   * @param senderid    der Absender der Botschaft
   * @param documentid  die ID des zu uebersetzenden Dokuments
   * @param error       (Fehler-)Meldungen des Compilers
   */
  void msgCommandDocumentCompile(VUserID senderid,
      VDocumentID documentid, String error);

  /**
   * Ausf"uhren eines Dokuments per Interpreter
   *
   * @param senderid    der Absender der Botschaft
   * @param command     Kommandozeile des Interpreter-Aufrufs
   */
  void msgCommandDocumentExecute(VUserID senderId,String command);
}
```

Auf eine vollständige Auflistung der einzelnen Methodenspezifikationen einer jeden Schnittstelle kann an dieser Stelle verzichtet werden.

8.4.2.3.2 Verwaltung von Nachrichten In der VitaminL-Software werden spzielle Objekte zur Lastverteilung (sog. *Dispatcher*) auf der Grundlage von

sogenannten *Threads* eingesetzt (vgl. Boger, 1999, Kap.2), um die Abarbeitung von Nachrichten von anderen Aktionen zu entkoppeln und damit zu einem flüssigeren Gesamtablauf beizutragen. Den Prototyp definiert die abstrakte Klasse `VADispatcher`, aus welcher diverse Ableitungen für den Versand (Klassen `VNetworkDispatcher` und `VServerConnectionSendDispatcher`), den Empfang und die weitere Verarbeitung (Klasse `VMessageDispatcher`) sowie die Protokollierung (Klasse `VLogDispatcher`) resultieren. Diese Dispatcher greifen auf Warteschlangen (*Queues*; (vgl. Balzert, 1999, S.397f)) zurück, die die Nachrichten nach dem FIFO[6]-Prinzip verwalten, wodurch die zeitliche Reihenfolge der einzelnen Nachrichten zueinander erhalten bleibt.

8.4.2.4 Phasen der Client-Server-Kommunikation

Die Kommunikation zwischen Client und Server besteht aus mehreren aufeinander folgenden Phasen, in denen jeweils Nachrichten versendet (und auf der Gegenseite empfangen) werden, um sie anschließend ihrer Verarbeitung zuzuführen. Die Ergebnisse eines solchen Verarbeitungssschritts können weitere Nachrichten zur Folge haben, mit denen ebenso verfahren wird.

8.4.2.4.1 Versand und Empfang von Nachrichten Üblicherweise wird der erste Schritt einer Verarbeitungskette auf einem Client ausgelöst, beispielsweise durch den Anwender, der mit der Client-Applikation interagiert, indem er ein Quelltext-Dokument editiert, mit anderen Teammitgliedern kommuniziert oder eine beliebige andere Aktion innerhalb der VitaminL-IDE ausführt. Als Folge davon wird ein geeignetes Nachrichtenobjekt erzeugt und an die Netzwerkschicht des Clients durchgereicht, wo es über die Netzwerkverbindung an die empfangende Gegenseite geschickt wird.

Auf der Empfängerseite wird eine eingehende Nachricht entgegengenommen, zusammen mit einer Referenz auf das verarbeitende Objekt (den Server oder Client bzw. die entsprechende Steuerungsschicht) in eine Art Container-Objekt eingepackt und an einen Dispatcher zur weiteren Verarbeitung übergeben. Der nachfolgende (stark gekürzte) Code (s. 8.2) soll diese prinzipielle Arbeitsweise verdeutlichen.

Beispiel 8.2: Empfang eines Nachrichtenobjekts auf dem Server

```
// Uebergib das Nachrichtenobjekt msg an den Protokollmechanismus log
log.log(msg);

// Packe die Nachricht msg zusammen mit ihrem Verarbeiter this
// in einen Nachrichtencontainer evt vom Typ VMessageEvent
VMessageEvent evt = new VMessageEvent(msg, this);

// Fuege den Nachrichtencontainer evt
// an das Ende der Warteschlange dispatcher an
```

[6] First In, First Out

```
dispatcher.send(new VMessageEvent(msg, this));

// Fertig - bis zur naechsten eintreffenden Nachricht
```
■

Auf diese Weise ist der Empfänger in der Lage, auch eine große Anzahl nahezu zeitgleich eintreffender Nachrichten entgegenzunehmen.

8.4.2.4.2 Verarbeiten von Nachrichten Die weitere Verarbeitung wird von einem Dispatcher geregelt, der in einem Thread – und damit zeitlich entkoppelt von der Aufnahme neuer Nachrichten – die eingegangenen Nachrichten und ihre jeweiligen Verarbeiter aus den Containern extrahiert. Im Folgeschritt wird jede Nachricht angewiesen, sich durch ihren Verarbeiter verarbeiten zu lassen. Die prinzipielle Arbeitsweise dieses Dispatchers kann dem folgenden Code-Extrakt entnommen werden.

Beispiel 8.3: Verarbeitung von Nachrichten durch einen Dispatcher

```
// Endlosschleife (solange der Dispatcher existiert)
while (true)
{
  // Falls die Warteschlange queue nicht leer ist...
  if( false == queue.isEmpty() )
  {
    // ...wird das vorderste Objekt entfernt...
    VMessageEvent evt = queue.remove();
    // ...und verarbeitet:
    // - extrahiere die Nachricht aus dem Container
    VMessage msg = evt.getMessage();
    // - extrahiere den zug. Verarbeiter aus dem Container
    VIProcessor proc = evt.getProcessor();
    // - starte die eigentliche Verarbeitung der Nachricht
    //   und uebergib mit der Botschaft process die Referenz
    //   auf den Prozesser vom Typ VIProcessor
    msg.process(proc);
  } //  Ende von if
}  //  Ende von while
```
■

Wird nun ein Nachrichtenobjekt angewiesen, sich zu verarbeiten, so wird dessen Operation `process()` (s. Kap.8.4.2.1.2) aufgerufen und der zugehörige Prozessor an die Operation übergeben. Diese in der Basisklasse `VMessage` abstrakt spezifizierte Operation wird in allen nicht-abstrakten Nachrichtenklassen mit einer geeigneten Implementierung versehen, in der zunächst der Typ des übergebenen Prozessors geprüft wird. Anschließend wird entsprechend des genauen Nachrichtentyps (s. Kap.8.4.2.1.1) die zugehörige Operation des Prozessors mit allen für die weitere Verarbeitung notwendigen Daten aufgerufen. Anhand der Nachrichtenklasse `VDocumentCommandMessage` soll dieser letzte Verarbeitungsschritt exemplarisch erläutert werden. Dazu enthält das nachfolgende Beispiel einen Auszug aus dem Code dieser Klasse, der nur wesentliche Anweisungen enthält und aus Gründen der Übersichtlichkeit auf Zusätze wie Fehlerbehandlungsmaßnahmen verzichtet.

Beispiel 8.4: Verarbeitungsroutine der Klasse `VDocumentCommandMessage`

```
void process(VIProcessor processor)
{
  // Ueberpruefe den konkreten Datentyp des (potentiellen) Verarbeiters
  if( processor instanceof VIDocumentCommandProcessor == false )
  {
    // FEHLER: FALSCHER VERARBEITER-TYP!
    return;
  }

  // Wandle den allgemeinen Verarbeiten in den tatsaechlichen Verarbeiter um
  VIDocumentCommandProcessor proc = (VIDocumentCommandProcessor)processor;

  // Bestimme nun den genauen Typ der Botschaft anhand des Attributs type
  switch ( type )
  {
    // Nachrichtentyp fuer Dokumentbefehl CLOSE (Dokument schliessen)
    case V_COMMAND_DOCUMENT_CLOSE:
           proc.msgCommandDocumentClose(senderId, documentID);
           break;

    // Nachrichtentyp fuer Dokumentbefehl COMPILE (Dokument uebersetzen)
    case V_COMMAND_DOCUMENT_COMPILE:
           proc.msgCommandDocumentCompile(senderId, documentID,
                                          argument.toString());
           break;

    // Nachrichtentyp fuer Dokumentbefehl EXECUTE (Ausfuehren per Interpreter)
    case V_COMMAND_DOCUMENT_EXECUTE:
           proc.msgCommandDocumentExecute(senderId, argument.toString());
           break;

    // Unbestimmter Nachrichtentyp
    default:
           // FEHLER
  }//  Ende von switch
}
```

■

Die möglichen Nachrichtentypen sind als Konstante in der jeweiligen Nachrichtenklasse (in vorigem Beispiel: `VDocumendCommandMessage`) definiert und werden dem Attribut `type` des Nachrichtenobjekts bei dessen Erzeugung zugewiesen. Die weitere Verarbeitung ist im jeweiligen Prozessor festgelegt.

8.4.3 Codierung von Handlungen

Aufbauend auf diesem Konzept nachrichtenbasierter Kommunikation wird der in Kapitel 8.2 behandelte Ansatz strukturierter Kommunikation in ein gesamtheitliches Modell zur Beschreibung sämtlicher Aktivitäten der Anwender im VitaminL-System überführt mit dem Ziel, ein Analyseverfahren zu entwickeln, das anhand der Aktivitäten eines Benutzers Schlüsse bezüglich Problemsituationen und angemessene Unterstützung ziehen kann.

8.4.3.1 Strukturierte Kooperation

Wie in dem vorangegangenen Kapitel 8.4.2 zu sehen ist, wird nicht nur die Kommunikation der Teilnehmer einer VitaminL-Sitzung untereinander mittels Nachrichtenklassen und -objekten realisiert, sondern darüber hinaus auch die Zusammenarbeit der Teilnehmer sowie alle damit verbundenen Tätigkeiten einzelner Mitglieder. Dies umfasst typische Tätigkeiten der Erstellung und Bearbeitung von Quelltexten genauso wie das Freigeben und Anfordern von Dokumenten (s. Kap.8.3.1) sowie weitere, für die Programmierung in Java spezifische Operationen. Diese Tätigkeiten verteilter, synchroner Programmierung lassen sich wie folgt strukturieren:

- **Dateioperationen**: Zu diesen Operationen zählen alle Tätigkeiten, die sich direkt auf Dateien als Container von Quelltext-Dokumenten auswirken. Dies umfasst neben dem Erzeugen neuer Dokumente auch das Laden und Speichern von Dokumenten sowie das Umbenennen und das Schließen.

- **Editoroperationen**: Diese Kategorie umfasst Tätigkeiten der Programmierung, die sich auf den Inhalt von Quelltext-Dokumenten auswirken. Neben dem Einfügen und Löschen von Text fällt in diese Kategorie auch das Ändern von Attributen eines Texts sowie das Markieren von Textbereichen innerhalb eines Dokuments.

- **Navigation**: Die Navigationstätigkeiten beinhalten das Navigieren innerhalb eines Quelltext-Dokuments, bei dem der Anwender die Position des Text-Cursors ändert, sowie das Navigieren zwischen Dokumenten, das in der Regel durch Auswahl eines anderen Dokuments ausgelöst wird.

- **Dokumentenaustausch**: Neben den bereits in Kapitel 8.3.1 vorgestellten Operationen zum Anfordern eines Dokuments durch einen Anwender zum Zwecke der Bearbeitung und der Freigabe von Dokumenten durch deren Besitzer fällt auch die Möglichkeit, einem Benutzer auch ohne dessen Zustimmung ein Dokument zu entziehen - diese Operation ist aber derzeit nur Benutzern mit besonderem Status (Tutoren und Administratoren) zugänglich und findet daher in der Regel kaum Anwendung.

- **Java-Operationen**: In der VitaminL-Software sind als typische Tätigkeiten der Java-Programmierung derzeit der Aufruf des Java-Compilers zum Übersetzen von in Quelltext-Dokumenten definierten Java-Klassen sowie der Aufruf des Java-Interpreters zum Zwecke der Ausführung von Java-Applikationen. Die Integration des Java-Debuggers für eine verbesserte Fehlersuche sind zwar geplant, aber derzeit noch nicht umgesetzt.

Es ergibt sich eine Ergänzung der strukturierten Kommunikation um eine Menge von Kooperationstätigkeiten, die sämtliche Handlungsmöglichkeiten umfassen, die während der gemeinsamen, synchronen Programmierung in Java mit dem VitaminL-System von den Teilnehmern ausgelöst werden können. Fasst man nun – in Anlehnung an Austins Sprechakte (s. Austin, 2002; Searle, 1983) – auch

die Kommunikation als Handlung auf, so resultiert daraus eine integrierende Sichtweise, in Folge derer die Kommunikation als Speziallfall der Kooperation verstanden werden kann. In der vorliegenden Arbeit wird dies als *strukturierte Kooperation* bezeichnet.

8.4.3.2 CLS++

Aus der Ausdehnung der in Kapitel 8.2.1 eingeführten Kategorisierung von Kommunikation auf sämtliche Tätigkeiten der Zusammenarbeit im VitaminL-System ergibt sich konsequenterweise auch eine Erweiterung des Ansatzes zur Kommunikationscodierung. Aufbauend auf dem in Tabelle 8.2 definierten Code werden auch die im vorigen Kapitel skizzierten Tätigkeiten einbezogen. Als Resultat ergibt sich eine Codierung sämtlicher Elemente der strukturierten Kooperation, deren konkrete Umsetzung mittels der zuvor beschriebenen Nachrichtenklassen (s. Kap.8.4.2.1 und 8.4.2.2) erfolgt. Insbesondere mit dem in der Oberklasse `VMessage` definierten Attribut `code` (vom Datentyp `int`) wird die Codierung modelliert. Die den einzelnen Elementen der strukturierten Kommunikation zugeordneten Codes sind in der folgenden Tabelle 8.4 zusammengefasst. Dabei wurde zugunsten besserer Lesbarkeit auf eine Wiederholung der bereits in Tabelle 8.2 genannten Elemente verzichtet. Eine tabellarische Zusammenfassung aller Elemente der strukturierten Kommunikation und Kooperation findet sich im Anhang A.1 in der Tabelle A.1.

Tab. 8.4: Codierung der Elemente der strukturierten Kooperation

Kategorie	**Element**	**Code**
I. Dateioperationen	Erzeugen	39
	Laden	40
	Speichern	41
	Umbenennen	42
	Schließen	43
J. Editoroperationen	Einfügen	37
	Löschen	38
	Ändern	50
	Markieren	52
K. Navigation	Navigation in einem Dokument	35
	Navigation zwischen Dokumenten	36
L. Dokumentenaustausch	Anfordern	44
	Freigeben	45
	Entziehen	46
M. Java-Operationen	Übersetzen mit Compiler	47
	Ausführen mit Interpreter	48

In Anlehung einerseits an die *Collaborative Learning Skills* (CLS) als Basiskonzept der strukturierten Kommunikation sowie andererseits an die objektorientierte

Programmierung als Anwendungsszenario wird diese Codierung in der vorliegenden Arbeit als *CLS++* bezeichnet werden. Der Inkrement-Operator `++`, der schon bei der Namensgebung der objektorientierten Programmiersprache *C++* Verwendung fand, um *„den evolutionären Charakter der Sprachentwicklung aus C heraus"* (Stroustrup, 1998, S.11) zu verdeutlichen, und der ebenfalls in der Programmiersprache Java wiederzufinden ist, soll auch hier die Weiterentwicklung des ursprünglichen Konzepts der strukturierten Kommunikation gemäß CLS kennzeichnen. Da CLS++ als echte Erweiterung von CLS verstanden wird und vornehmlich im weiteren Verlauf dieser Arbeit verwendet werden wird, können die Begriffe *CLS++-Codes*, *CLS-Codes* und *Codes* zu Gunsten einer besseren Lesbarkeit synonym verwendet werden. Wo eine Differenzierung notwendig sein sollte, wird dieses explizit erfolgen.

8.4.4 XML-basierte Nachrichtenprotokollierung

Wie in 8.2 zu sehen ist, wird jede Nachricht beim Empfang auf dem Server vor ihrer eigentlichen Verarbeitung mittels eines Protokollmechanismus aufgezeichnet. Dabei werden eingehende Nachrichtenobjekte an ein spezielles Protokollierungsobjekt des Servers übergeben, welches aus einem Nachrichtenobjekt die zu protokollierenden Informationen liest und als XML in eine Protokolldatei schreibt. Auf diese Weise werden alle Aktivitäten der Gruppenmitglieder während einer VitaminL-Sitzung erfasst und somit jede Sitzung vollständig protokolliert.

8.4.4.1 Erzeugung von XML-Logfiles

An der Protokollierung von Nachrichtenobjekten im XML-Format sind – neben den bereits in Kapitel 8.4.2 behandelten Nachrichtenklassen – weitere Klassen und deren Instanzen beteiligt.

8.4.4.1.1 Die Schnittstellenklasse `VILogger` In der Schnittstelle `VILogger` wird der allgemeine Protokollmechanismus mittels zweier Operationen spezifiziert:

- `void log(Object o)`:
  Mit dieser Operation wird ein beliebiges Objekt an den Protokollmechanis zwecks Protokollierung übergeben.

- `void close()`:
  Der Aufruf dieser Methode beendet den Protokollmechanismus und ermöglicht die Ausführung von Aufräumarbeiten wie das Schließen von Kommunikationskanälen oder die Terminierung aktiver Objekte (Threads).

Eine Operation zum Öffnen (bspw. `void open()`) als Gegenstück zur Operation `void close` wird nicht explizit spezifiziert. Vielmehr werden alle Aktionen, die für die Initialisierung eines bestimmten Protokollmechanismus erforderlich sind, in der

jeweiligen Implementierung der Schnittstelle `VILogger` ausgeführt. Dies geschieht üblicherweise bei der Objekterzeugung im Konstruktur der entsprechenden Klasse. Auf diese Weise lässt sich ein allgemeingültiger Protokollmechanismus beschreiben, der in unterschiedlichen Ausprägungen realisiert werden kann und der daher bei Initialisierung unterschiedliche Parameter benötigt. Denkbar ist die Erfassung von Nachrichtenobjekten in Protokolldateien genauso wie eine dauerhafte Speicherung in einer Datenbank.

8.4.4.1.2 Protokolldateien mit der Klasse `VLogger` Diese Klasse implementiert die Schnittstelle `VILogger` und stellt einen Protokollmechanismus, zur Verfügung, der auf Dateien basiert, die auf dem VitaminL-Server in dessen lokalen Dateisystem erzeugt und hinterlegt werden. Dem Konstruktor wird dazu ein beliebiger Name (vom Typ `String`) übergeben, der die zu erstellende Protokolldatei näher beschreibt: Dieser Name kann ein Gruppenname sein (für die Protokollierung von Nachrichten einer Gruppensitzung) oder aber auch einen anderen Inhalt besitzen (so wird bspw. der Name `vitaminl` verwendet, um Informationen des Servers während des laufenden Betriebs zu erfassen). Aus diesem Namen wird – zusammen mit einem Verzeichnisnamen und dem aktuellen Zeitstempel (bestehend aus Datum und Uhrzeit zum Zeitpunkt des Konstruktoraufrufs) – der Dateiname der Protokolldatei generiert. Mit diesem Dateinamen wird daraufhin ein spezieller Thread vom Typ `VLogDispatcher` erzeugt und gestartet, so dass an dieser Stelle eine zeitliche Entkopplung des eigentlichen Schreibvorgangs im Zuge der Protokollierung von Objekten ermöglicht wird.

Die Hilfsklasse `VLogDispatcher` Als eine Ableitung der Klasse `VADispatcher` (s. Kap.8.4.2.3.2) stellt die Klasse `VLogDispatcher` eine thread-gesteuerte Warteschlange für die Aufnahme und weitere Verarbeitung von zu protokollierenden Einträgen (vom Typ `String`) bereit, mit deren Hilfe diese Klasse das zeitlich-entkoppelte Schreiben in eine Protokolldatei realisiert.

Beim Erzeugen einer Instanz dieser Klasse wird der Dateiname per Konstruktor an die Instanz übergeben, woraufhin die zugehörige Datei zum Schreiben geöffnet und mit einer einleitenden XML-Deklaration (<?xml version='1.0' encoding='ISO-8859-1'?>) sowie einem Start-Tag (<logfile>), gefolgt von einem XML-Element mit einer Versionsangabe (<VERSION>3.00.3</VERSION>) beschrieben wird.

Nach dieser Vorbereitung können XML-konforme (`String`-)Objekte an die Warteschlange des Dispatchers angefügt werden, die unter Beibehaltung der zeitlichen Reihenfolge ihres Eintreffens ebenfalls in die Protokolldatei geschrieben werden. Das Beenden der Protokollierung durch den Aufruf der Operation `close()` des umgebenden `VLogger`-Objekts beendet auch den Protokoll-Dispatcher, infolgedessen die Warteschlange komplett abgearbeitet und ein abschließendes End-Tag (</logfile>) geschrieben wird, so dass anschließend der Ausgabestrom zur Protokolldatei geschlossen wird. Da alle zu protokollierenden Einträge ebenfalls XML-konform sind, ergeben sich mit dem abschließenden End-Tag (</logfile>)

korrekte, vollständige XML-Dokumente, die von einem XML-Parser (wie SAX oder DOM) gelesen, auf Gültigkeit geprüft und weiterverarbeitet werden können.

Die Operation `void log( Object o)` Diese Operation stellt die Implementierung der Schnittstelle `VILogger`, die aus jedem an sie übergebenen Objekt eine geeignete Textdarstellung liest, diese in ein XML-Element überführt, mit einem Zeitstempel versieht und an den Protokoll-Dispatcher weiterreicht. Gemäß Spezifikation ist diese Operation für die Protokollierung beliebiger Objekte geeignet: Die Klasse `Object` als Basisklasse sämtlicher Java-Klassen definiert die Operation `toString()`, die *„eine lesbare Repräsentation des Objekts"* (Schiedermeier, 2005, S.264) vom Typ `String` liefert. Da diese Operation in der Klasse `Object` nur eine minimale Funktionalität besitzt, erfolgt in Ableitungen üblicherweise eine angepasste Redefinition.

Darüber hinaus signalisiert die Schnittstelle `VILoggable` die Bereitschaft von Klassen (und deren Objekten), den Protokollmechanismus durch die Bereitstellung von spezifischeren Textrepräsentationen besser zu unterstützen als solche Klassen, die diese Schnittstelle nicht implementieren. Dieser Umstand wird auch in der Implementierung der Operation `log` in der Klasse `VLogger` berücksichtigt: Es erfolgt eine Unterscheidung zwischen Objekten, deren Klassen die Schnittstelle VILoggable implementieren, und sonstigen Objekten. Der folgende Code-Auszug enthält die Definition der Operation `log()`.

Beispiel 8.5: Objektprotokollierung: Operation `VLogger::log()`

```
void log( Object o )
{
  // Implementiert das Objekt die Schnittstelle VILoggable?
  if( o instanceof VILoggable )
  {
    // Ja, dann koennen wir auf dessen Log-String zugreifen
    // und diesen zur Protokollierung verwenden
    writeToLog( "<" + o.getClass().getName() + ">"
              + ((VILoggable)o).getLogString()
              + "</" + o.getClass().getName() + ">" );
  }
  // Nein, VILoggable ist nicht implementiert, daher...
  else
  {
    // ...erfolgt Protokollierung weniger detailiert
    writeToLog( "<general><" + o.getClass().getName() + ">"
              + o.toString();
              + "</" + o.getClass().getName() + "></general>" );
  }
}
```

■

In jedem Fall erzeugt die Operation eine XML-konforme Zeichenkette, die an die Operation `writeToLog()` weitergereicht wird. Dort wird sie ergänzt um einen Zeitstempel und zusammen mit diesen in ein neues XML-Element (`<log>...</log>`) gepackt. Dieses wird abschließend an den Protokoll-Dispatcher übergeben, der den eigentlichen Schreibvorgang in dessen (XML-)Datei durchführt.

Beispiel 8.6: Objektprotokollierung: Operation `VLogger::writeToLog()`

```
void writeToLog( String t )
{
  // Erzeuge Start-Tag von umfassenden XML-Element
  StringBuffer logString = new StringBuffer( "<log>" );

  // Fuege Zeitstempel in XML-Format hinzu
   logString.append("<timestamp>"
                    + VTimestamp.getShortDate()
                    + "</timestamp>" );

  // Fuege zu protokollierenden (XML-konformen) Text hinzu
  logString.append( t );

  // Komplettiere gesamtes XML-Element mit End-Tag
  logString.append( "</log>" );

  // Reiche komplettes XML-Element an Protokoll-Dispatcher weiter
  logDispatcher.send( logString );
}
```

■

Nach diesem Prinzip lassen sich sämtliche Objekte unter Verwendung von XML in einer Datei abspeichern.

8.4.4.2 XML-Transformation von Nachrichten

Nachdem in den vorangegangenen Ausführungen das Prinzip der Erzeugung von XML-Protokolldateien bereits erläutert wurde, soll in diesem Kapitel die Transformation von Nachrichtenobjekten in das XML-Format genauer beleuchtet und anhand einer konkreten Nachrichtenklasse exemplarisch demonstriert werden.

8.4.4.2.1 XML-Transformation von Klassen Wie bereits im Beispiel 8.5 zu sehen ist, wird bei der Protokollierung eines (Nachrichten-)Objekts auf dessen Klassenname zugegriffen, um diesen für das Start- und End-Tag eines neuen XML-Element zu verwenden. Den Inhalt des XML-Elements liefert der Aufruf der Operation `getLogString()`, deren Implementierung in der Klasse `VMessage` als Basisklasse sämtlicher Nachrichtenklassen auf die Operation `toXMLString()` zurückgreift und deren Ergebnis (vom Typ `String`) zurückliefert. Da aber `toXMLString()` wie die Klasse `VMessage` selbst abstrakt ist, muss die Operationen in allen nicht-abstrakten Ableitungen jeweils geeignet implementiert werden. Auf diese Weise kann jede Nachrichtenklasse einen Mechanismus zur Erzeugung von spezifischen Textrepräsentationen für ihre Objekte bereitstellen. Es ergibt sich somit eine Transformation von Nachrichtenobjekten in XML-Elemente.

Die folgende Tabelle 8.5 fasst dies für eine kleine Auswahl von nicht-abstrakten Nachrichtenklassen zusammen.

Tab. 8.5: Nachrichtenklassen und XML-Elemente (Auswahl)

Nachrichtenklasse	**XML-Element**
`VAccountMessage`	<vitaminl.message.VAccountMessage> ... </vitaminl.message.VAccountMessage>
`VAnalyzeMessage`	<vitaminl.message.VAnalyzeMessage> ... </vitaminl.message.VAnalyzeMessage>
`VCommunicationMessage`	<vitaminl.message.VCommunicationMessage> ... </vitaminl.message.VCommunicationMessage>
`VDocumentErrorMessage`	<vitaminl.message.VDocumentErrorMessage> ... </vitaminl.message.VDocumentErrorMessage>
`VDocumentUpdateMessage`	<vitaminl.message.VDocumentUpdateMessage> ... </vitaminl.message.VDocumentUpdateMessage>
`VFeedbackMessage`	<vitaminl.message.VFeedbackMessage> ... </vitaminl.message.VFeedbackMessage>
`VSystemMessage`	<vitaminl.message.VServerMessage> ... </vitaminl.message.VServerMessage>

8.4.4.2.2 XML-Transformation von Attributen Über die Operation `toXMLString()` definiert jede Nachrichtenklasse die Erzeugung von XML-konformen Textrepräsentationen ihrer Objekte. Dies erfolgt in Form einer Abbildung relevanter Attribute (samt ihrer jeweiligen Werte) auf geeignete XML-Elemente. Da das Prinzip dieser Transormation für alle Nachrichtenklassen identisch ist, soll es an dieser Stelle genügen, dies exemplarisch für eine konkrete Nachrichtenklasse zu demonstrieren. Es wird zu diesem Zwecke die Klasse `VCommunicationMessage` gewählt.

Wie im nachfolgenden Beispiel-Code zu sehen ist, werden alle für die Protokollierung relevanten Attribute sukzessive auf XML-Elemente abgebildet und mit einem umgebenden Tag zu einem XML-Element zusammengefasst.

Beispiel 8.7: Operation `toXMLString()` der Klasse `VCommunicationMessage`

```
String toXMLString()
{
  // Start-Tag erzeugen = XML-Element oeffnen
  String xml = "<VCOMMUNICATIONMESSAGE>"
    // XML-Element fuer Absender-Kennung anhaengen
    + VLogger.createXML_Tag( "SENDERID" , getSenderId() )
```

```
    // XML-Element fuer Nachrichtentyp anhaengen
    + VLogger.createXML_Tag( "TYPE" , getTypeString( typeString ) )
    // XML-Element fuer CLS++-Code anhaengen
    + VLogger.createXML_Tag( "CODE" , new Integer( getCode() ) )
    // XML-Element fuer Empfaengerkennung
    + VLogger.createXML_Tag( "RECEIVERID" , getReceiverId() )
    // XML-Element fuer Inhalt der Botschaft anhaengen
    + VLogger.createXML_Tag( "CONTENT" , "<![CDATA[" + getContent() + "]]>" )
  // End-Tag anhaengen = XML-Element schliessen
    + "</VCOMMUNICATIONMESSAGE>";

  // XML-Element als Ergebnis zurueckliefern
  return xml;
}
```

Die in dieser Methode verwendete Operation `public static String createXML_Tag( String element , Object value )` ist eine statische Operation der Klasse `VLogger` Mit ihr kann ein beliebiges Objekt (Paramter `Object value`) in ein XML-Element transformiert werden, dessen Name (Tag) ebenfalls als Paramter (`String element`) übergeben wird.

8.4.4.2.3 XML-Transformation von Objekten An einem abschließenden Beispiel soll dieser für die im Rahmen dieser Arbeit durchgeführte Analyse relevante Mechanismus illustriert werden.

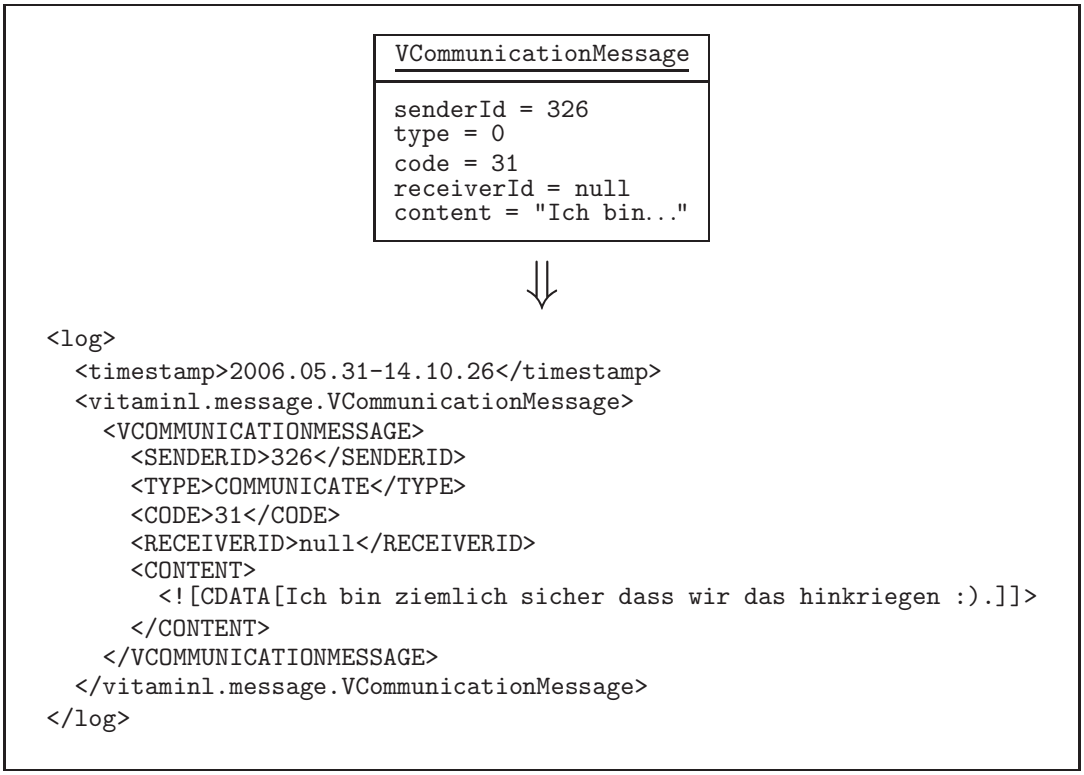

Abb. 8.23: Vom Objekt zum XML-Protokolleintrag

Darstellungsgegenstand dieses Beispiels sei ein Kommunikationsbeitrags eines

Teilnehmers an seine Teammitglieder. Das entsprechende Objekt sowie der resultierende XML-Eintrag in einer Protokolldatei sind in Abbildung 8.23 dargestellt. Im oberen Teil der Abbildung ist das Beispielobjekt vom Typ `VCommunicationMessage` mitsamt seinen für die Protokollierung relevanten Attributwerten als UML-Objektdiagramm dargestellt.

Im Zuge der zuvor beschriebenen Protokollierung in einer XML-Protokolldatei wird das Objekt in das darunter abgebildete XML-Element überführt. Wie in der Abbildung des Beispiels deutlich zu erkennen ist, bleiben sämtliche Informationen des Objekts erhalten. Dies stellt eine für die weitere Verwendung der Protokolldateien wichtige Eigenschaft dar.

8.4.4.3 Anwendungen

Die Generierung von Protokolldateien dient primär der vollständigen Erfassung von Sitzungen inklusive aller während einer Sitzung durchgeführten Aktionen sämtlicher Teilnehmer. Nach Beendigung einer Sitzung kann diese unter Zuhilfenahme der Protokolle einer weiteren Verarbeitung zugeführt werden.

XML kommt in diesem Zusammenhang als Dateiformat zum Einsatz, weil es als wohldefiniertes Dateiformat im ASCII-Code gleichsam menschen- wie maschinenlesbar ist. Speziell der Einsatz von XML-Parsern vereinfacht das maschinelle Lesen stark und erleichtert infolgedessen auch die maschinelle Auswertung der in einer Protokolldatei enthaltenen Einträge.

Dieser Umstand ist für die in den Kapiteln 10 und 11 vorgestellten Verfahren relevant, wirkt sich aber auch günstig auf die nachfolgenden Anwendungen aus.

8.4.4.3.1 Wiedergabe von Sitzungen Mit einer speziellen Erweiterung der VitaminL-Software, dem sogenannten *Logfile-Analyzer*, können Sitzungen anhand der Protokolle wiederholt werden. Durch serielles Auslesen der XML-Elemente ist es möglich, die zuvor gespeicherten Objekte zu rekonstruktieren und die Sitzung inhaltsgetreu ohne Beisein der realen Teilnehmer erneut durchzuführen (s. Abb.8.24).

Nach Auswahl einer Protokolldatei werden automatisch alle beteiligten Anwender an einem lokal gestarteten Server angemeldet und zugehörige Client-Fenster geöffnet (Bereiche *Client 1* - *Client 4*). Die von den Teilnehmern ausgelösten Aktionen werden in einem weiteren Fenster gemäß ihrer zeitlichen Reihenfolge dargestellt (Bereich *Benutzeraktionen*) und können über Bedienelemente (Bereich *Bedienelemente*) nacheinander ausgelöst werden, um somit den Verlauf der ursprünglichen Sitzung zu simulieren. Zu jeder Nachricht werden in einem weiteren Teilfenster Zusatzinformationen ausgegeben (Bereich *Informationsfenster*).

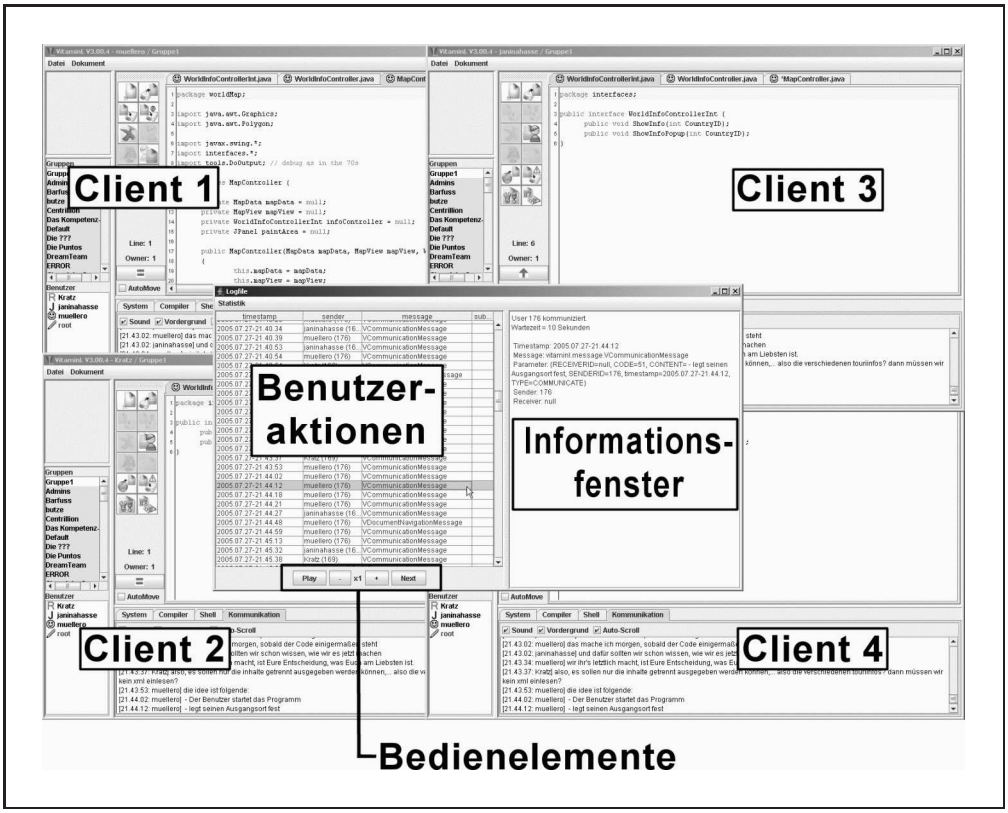

Abb. 8.24: Wiedergabe einer protokollierten Sitzung mit dem Logfile-Analyzer (Screenshot)

Ähnlich wie bei einem Video- oder einem DVD-Wiedergabegerät kann die Geschwindigkeit der Wiedergabe durch den Anwender beeinflusst werden, so dass neben einer schnellen oder langsamen Wiedergabe (Zeitraffer respektive Zeitlupe) auch jederzeit angehalten (Pause) oder gar abgebrochen (Stop) werden kann. Somit können durch Wiederholungen von Sitzungen auch im Nachhinein gezielt bestimmte Aspekte beobachtet werden, ohne die originäre Sitzung in irgendeiner Weise zu beeinflussen und damit unter Umständen zu verfälschen.

8.4.4.3.2 Bereitstellung einer Wissensbasis Ein weiteres Anwendungsgebiet stellt die Auswertung kompletter Sitzungen in Kombination mit anderen Datenbeständen dar. Im Hinblick auf die in Kapitel 7.3.4 geäußerten Fragestellungen repräsentieren die XML-Dateien mit den enthaltenen Sitzungsdaten nur eines von mehreren zu betrachtenden Elementen. Mittels eines Datenbank-Management-Systems (DBMS) werden zu diesem Zwecke sämtliche für die geplante Auswertung relevanten Informationen wie beispielsweise Sitzungsdaten und Rollenprofile in eine Wissensbasis integriert.

Das Ziel besteht darin, eine Datenbasis bereitzustellen, auf die mit Standard-Werkzeugen zugegriffen werden kann. Dieser Zugriff kann in Form von SQL-Abfragen aus einer entsprechenden Anwendung (wie beispielsweise der `mysql`-Shell oder dem Programm *SQLyog*) heraus erfolgen. Ferner können spezielle Anwendungen wie Tabellenkalkulationen (*MS Excel*, *OpenOffice Calc* etc.) oder Statistik-

Programme wie *SPSS* eingesetzt werden, der Zugriff kann aber auch direkt aus Java-Programmen mittels der JDBC-Schnittstelle unter Verwendung eines geeigneten Datenbank-Treibers (hier: MySQL-Treiber) durchgeführt werden. Die Grundlage der Wissensbasis bildet das bereits für die Verwaltung von Benutzer- und Gruppendaten verwendete DBMS (s. Kapitel 8.4.1.1.1), das geeignet erweitert wird.

Inhalt und Struktur der Wissensbasis Im Zuge der Arbeiten am VitaminL-Projekt wurde die ursprüngliche Datenbasis (s. Kap.8.4.1.1.1) schrittweise erweitert und beinhaltet zum derzeitigen Zeitpunkt einerseits Informationen über die Rollenausprägungen teilnehmender Benutzer in Form von Rollenprofilen auf der Basis des Rollenmodells von Spencer & Pruss (s. Kap.2.5.5), andererseits auch ausgewählte Sitzungsdaten als Extrakt aus den XML-Protokolldateien. Die Struktur der Wissensbasis ist in der folgenden Abbildung 8.25 auszugsweise wiedergegeben.

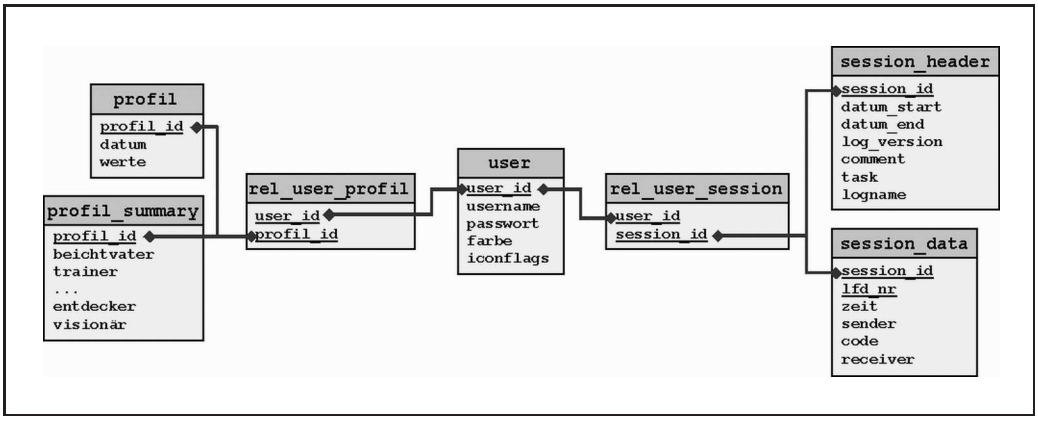

Abb. 8.25: Struktureller Aufbau der Wissensbasis
(Auszug)

Jeder Kasten in der Graphik repräsentiert eine Tabelle innerhalb der Wissensbasis, wobei der Name der Tabelle im oberen Teilfeld notiert ist und die zugehörigen Attribute (in Auszügen) dem darunter befindlichen Feld zu entnehmen sind. Unterstrichene Attribute stellen Schlüsselattribute (sog. *primary keys*) dar, anhand derer jeder Eintrag innerhalb einer Tabelle identifiziert werden kann. Beziehungen zwischen Tabellen werden in der vorliegenden Graphik in Form von Verbindungslinien notiert, deren Endpunkte diejenigen Attribute markieren, über die die Verbindung realisiert wird. Die Tabellen enthalten unter anderem folgende Information:

- Die Tabelle `user` enthält Informationen über teilnehmende Benutzer in Form von Benutzerkennung (Attribut `username`) und Passwort (`password`) sowie zugeordnete Schriftfarbe (`farbe`). Die Identifizerung erfolgt über das Schlüsselattribut `user_id`.
- Die Tabelle `profil` enthält Ergebnisse von Rollenfragebögen teilnehmender Benutzer auf der Grundlage des Rollenmodells von Spencer & Pruss. Je-

der Tabelleneintrag repräsentiert einen ausgefüllten Fragebogen und enthält neben den im Attribut `werte` zusammengefassten Einzelantworten auch das Datum der Befragung (`datum`). Das Schlüsselattribut `profil_id` dient der eindeutigen Erkennung jedes Fragebogens.

- Die Tabelle `profil_summary` beinhaltet Rollenprofile nach SPENCER & PRUSS (1995), die sich durch Verdichtung der Daten eines ausgefüllten Fragebogens ergeben. Pro Rolle existiert ein Attribut (`beichtvater`, `trainer` etc.) mit einer entsprechenden Rollenausprägung als Folge der Auswertung des zugehörigen Fragebogens, der anhand des Schlüsselattributs `profil_id` identifiziert werden kann.

- Die Tabelle `rel_user_profil` ist eine Relationentabelle und stellt über die Attribute `user_id` und `profil_id` die Beziehung eines Benutzers zu seinen Fragebogendaten her, indem zu einem beliebigen Benutzer (Tabelle `user`), der anhand des Attributs `user_id` identifiziert wird, über einen Eintrag in der Relationentabelle `rel_user_profil` eine zu dem Benutzer gehörende Profilkennung (Attribut `profil_id`) gesucht werden und die zugehörigen Profildaten aus den Tabellen `profil` und `profil_summary` gelesen werden.

- Die Tabelle `session_header` beinhaltet grundlegende Informationen einer VitaminL-Sitzung wie Anfangs- und Enddatum und -uhrzeit (Attribute `datum_start` und `datum_end`), den Versionsstand der Protokolldatei (`log_version`) und deren Dateiname (`logname`) sowie weitere Angaben bezüglich der Aufgabenstellung (`comment` und `task`). Jede Sitzung lässt sich anhand des Schlüsselattributs `session_id` identifizieren.

- Die Tabelle `session_data` beinhaltet alle Aktionen einer Sitzung (Attribut `session_id`), die von den Teilnehmern während der Sitzung ausgelöst wurden. Jede Aktion wird im Attribut `code` durch ihren zuvor vereinbarten Code (s. Kap.8.4.3) hinterlegt und in der Reihenfolge ihres Auftretens mit einer laufender Nummer (`lfdnr`) versehen. Desweiteren wird der Zeitpunkt ihres Auftretens (in Sekunden seit Sitzungsbeginn; Attribut `zeit`), die Benutzer-ID ihres Auslösers (`sender`) sowie die (optionale) Benutzer-ID eines dedizierten Empfängers (`receiver`) vermerkt. Das Attribut `content` enthält – sofern vorhanden – den eigentlichen Inhalt einer Benutzeraktion; dies ist insbesondere bei Kommunikationsnachrichten der Fall.

- Die Tabelle `rel_user_session` stellt die Verbindung zwischen einer Sitzung (Attribut `session_id`) und deren Teilnehmern (Attribut `user_id`) her.

Auf eine vollständige Beschreibung der Datenbasis wird an dieser Stelle zugunsten der Lesbarkeit verzichtet; der Anhang A.2 enthält weiterführende Detailinformationen.

Erfassung von Inhalten der Wissensbasis Während die Benutzerinformationen weitestgehend manuell erfasst werden, erfordert die Erfassung der Benutzer-

profile und der Sitzungsdaten aufgrund der vergleichsweise großen Datenmenge[7] den Einsatz geeigneter Software-Werkzeuge, um diese Informationen geeignet aufzubereiten und in die Wissensbasis einzupflegen. Zu diesem Zweck wurde eine spezielle Applikation entwickelt, die in Java geschrieben ist und über einen JDBC-MySQL-Treiber auf das verwendete MySQL-DBMS zugreift.

Neben der Erfassung von Rollenprofilen werden mit diesem Werkzeug hauptsächlich VitaminL-Sitzungen aus XML-Protokolldateien ausgelesen, gemäß der zuvor definierten Datenbankstruktur (s.o.) aufbereitet und mit SQL-INSERT-Anweisungen in die entsprechenden Tabellen eingefügt. Pro Sitzung wird in diesem Vorgang ein neuer Eintrag in der Tabelle `session_header` erzeugt, wobei zunächst eine neue Sitzungskennung (Schlüsselattribut `session_id`) für die eindeutige Identifikation einer Sitzung und aller ihr zugeordneten Informationen generiert wird.

Nach der Ermittlung weiterer sitzungsglobaler Daten (Datum und Uhrzeit von Anfang und Ende, Versionsinformationen etc.; s.o.) wird pro Benutzeraktion ein Eintrag in der Tabelle `session_data` erzeugt, das heißt XML-Elemente werden in Tabellenzeilen mit entsprechenden Attributwerten überführt. Bei dieser Transformation eines XML-Elements in einen Datenbank-Eintrag erfolgt – im Gegensatz zur Transformation von Nachrichtenobjekten in XML-Elemente – eine Selektion und damit der Wegfall von solchen Attributen, die für die spätere Verwendung nicht relevant sind.

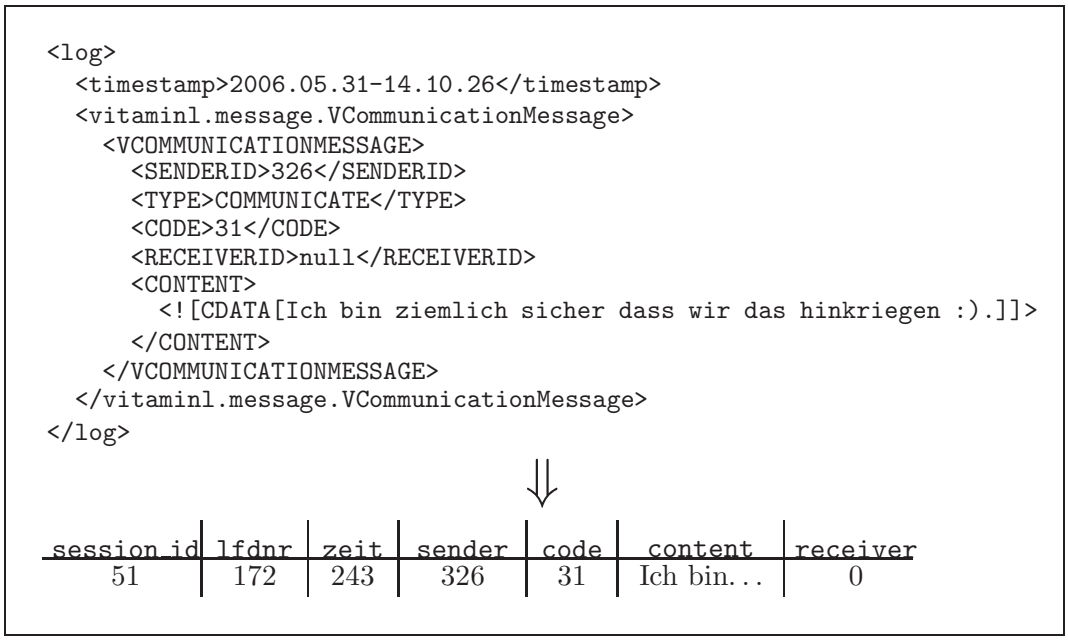

Abb. 8.26: Vom XML-Protokollelement zum Datenbankeintrag

Anhand des in Abbildung 8.26 dargestellten Beispiels wird die Transformation eines in einer Protokolldatei enthaltenen XML-Elements in einen Tabelleneintrag der Wissensbasis illustriert: Deutlich erkennbar ist die Überführung des

[7] Die XML-Protokolldatei einer typischen VitaminL-Sitzung von ca. 90 min Dauer ist in der Regel 1 - 2 MByte groß.

ursprünglichen Zeitstempels in eine Zeitangabe relativ zum Sitzungsbeginn sowie die unveränderte Übernahme der XML-Attribute `CODE`, `SENDERID`, `RECEIVERID` und `CONTENT` und deren Abbildung auf die entsprechenden Attribute der Tabelle `session_data`. Alle übrigen XML-Attribute sind für die spätere Analyse nicht relevant und werden daher gefiltert, das heißt sie entfallen komplett.

Auf diese Weise ist es möglich, alle im Zusammenhang mit VitaminL-Sitzungen erfassten Informationen auf einfache Weise in der Wissensbasis zu hinterlegen und mit Standard-Werkzeugen bei Bedarf Zugriff zu erlangen.

8.4.5 Zusammenfassung der Architektur

Die Architektur des CSCL-Systems zur Unterstützung von Lerngruppen bei der objektorientierten Programmierung in Java ergibt sich aus dem Zusammenspiel der vorgestellten Komponenten und Konzepte. Eine schematische Darstellung der Gesamtarchitektur ist in Abbildung 8.27 zu sehen.

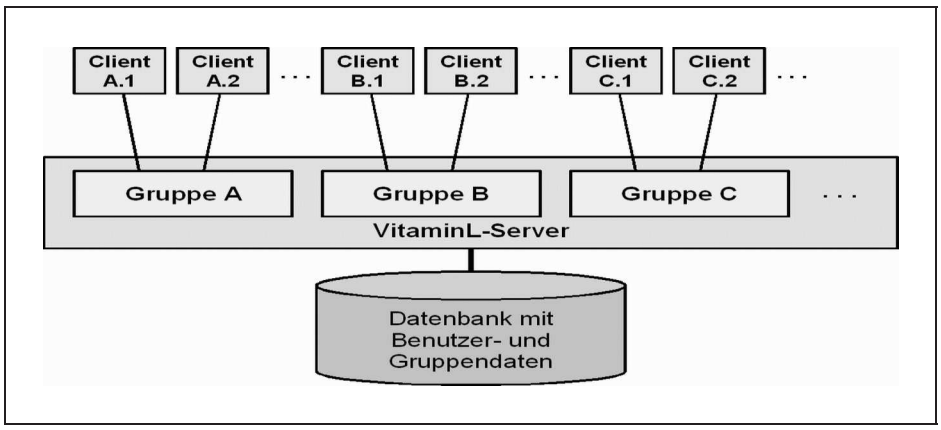

Abb. 8.27: Gesamtarchitektur des VitaminL-CSCL-Systems

Der VitaminL-Server als zentrale Instanz des CSCL-Systems ist einerseits an die bereits erwähnte Datenbank angebunden, in welcher Informationen über Benutzer und Benutzergruppen verfügbar sind, und besitzt andererseits Verbindungen zu sämtlichen angemeldeten Clients. Innerhalb des Servers sind alle angemeldeten Clients in Gruppen organisiert, die ihrerseits voneinander strikt getrennt werden, das heißt es gibt keine Verbindungen zwischen den Gruppen untereinander.

Die Architektur einer Gruppe (bzw. der zugehörigen Sitzung) kann der Abbildung 8.28 entnommen werden, in deren unterer Hälfte exemplarisch die server-seitig verwaltete Sitzung einer beliebigen Gruppe A zu sehen ist. Zentrale Komponenten einer solchen Gruppensitzung sind die Dokumentenverwaltung und die Kommunikationsverwaltung, die ergänzt werden durch einen Protokollierungsmechanismus sowie Informationen über alle an der aktiven Gruppensitzung angemeldeten Client-Rechner beziehungsweise deren zugehörige Verbindungen. Nicht eingezeichnet sind die Netzwerkschichten oder auch die diversen Dispatcher mitsamt ihren Warteschlangen, da diese innerhalb der Architektur eher unterstützenden Charakter besitzen.

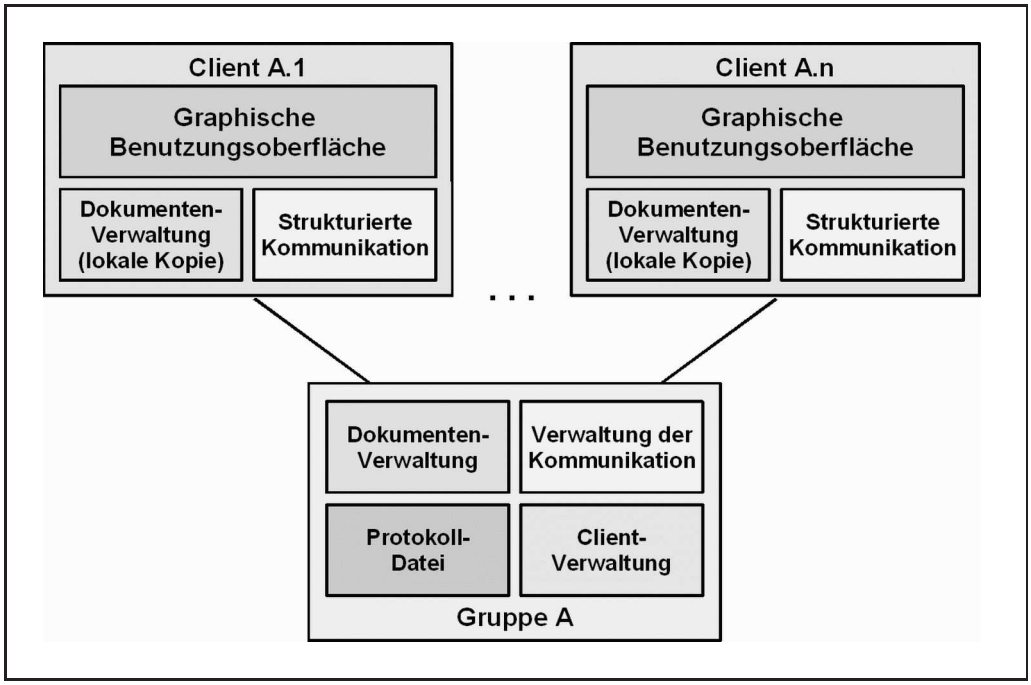

Abb. 8.28: Architektur einer VitaminL-Sitzung

Insgesamt stellt dieser Entwicklungsstand einen wichtigen Schritt dar auf dem Weg zu einem tutoriellen System, das in der Lage ist, virtuelle Teams beim synchronen, gemeinsamen Programmieren in Java und den daran gekoppelten Lernprozess *gezielt* und *aktiv* zu unterstützen.

8.5 Das VitaminL-System als Tutor-System

Gemäß des in Kapitel 6.3 erläuterten Architekturmodells intelligenter tutorieller Systeme sind ein Tutorenmodell, ein Lernermodell, ein Wissensmodell sowie eine Benutzerschnittstelle erforderlich (s. Abb.6.6). Im vorliegenden Fall ist bereits eine Benutzerschnittstelle für Interaktionen der Teilnehmer mit dem System (und damit systemvermittelt auch untereinander) vorhanden, auf die auch das zukünftige ITS zurückgreifen kann. Der Übergang von einer CSCL-Applikation zu einem ITS kann somit durch eine Ergänzung des bestehenden Systems um die noch fehlenden Modelle (für Lerner, Wissen und Tutor) vollzogen werden.

8.5.1 Komponenten tutorieller Unterstützung

Im Gegensatz zu traditionellen Tutorsystemen, deren Unterstützung sich meist auf einen Lernenden beschränkt, muss die Tutorkomponente des VitaminL-Systems in der Lage sein, eine virtuelle Lerngruppe als Ganzes zu unterstützen. Dies erfordert einige Modifikationen der in Abbildung 6.6 skizzierten Basisarchitektur.

8.5.1.1 Lernermodell

Anstelle eines Modells eines einzelnen Lernenden beinhaltet das Lernermodell des VitaminL-Tutors ein Teammodell, das Annahmen hinsichtlich Kenntnisstand und Fähigkeiten der gesamten Lerngruppe trifft und diese innerhalb der Tutorkomponente weiterreicht. Diesem Modell liegt die These zugrunde, dass bestimmte Funktionen von den Teilnehmern des Teams wahrgenommen werden müssen, um hinsichtlich der Aufgabenbearbeitung erfolgreich zu sein (s. Kap.2.5). Aus diesem Grunde wird zunächst pro Teilnehmer ein Lernendenmodell generiert, welches Informationen über dessen Kenntnisstand und Fähigkeiten enthält. Die Fähigkeiten lassen sich aus dem Rollenprofil des Teilnehmers ableiten, durch welches ja ausgedrückt wird, in welchem Umfang ein Teilnehmer zur Wahrnehmung bestimmter Teamfunktionen tendiert oder auch nicht tendiert. Besagte Teamfunktionen werden mittels eines Rollenmodells beschrieben, auf welches im Folgekapitel 9 ausführlich eingegangen wird.

Aus der Aggregation der Lernendenmodelle aller Teammitglieder ergibt sich das Teammodell, welches den Kenntnis- und Fähigkeitsstand des Teams zusammenfasst. Durch den Vergleich eines Teammodells mit einer idealen Teamzusammensetzung als eine Art Referenzmodell ergibt sich eine mögliche Differenz, die auf potentielle Defizite der Gruppe bezüglich der notwendigen Teamfunktionen hinweist. Dies wird an das Tutormodell weitergereicht, wo es bei der Generierung von Support Berücksichtigung findet.

Die Ermittlung der Lernendenmodelle als Basiselemente des Teammodells erfolgt auf Basis der Interaktionen der Teilnehmer mit dem System: Ansetzend an der Benutzerschnittstelle werden sämtliche Aktionen eines Teilnehmers – analog zur Protokollierung in Log-Dateien (s. Kap.8.4.2.4 und Kap.8.4.4) – erfasst und analysiert mit dem Ziel, Rückschlüsse hinsichtlich der Wahrnehmung von Teamfunktionen zu ziehen und somit ein Rollenprofil zu generieren. Die in diesem Zusammenhang durchgeführten Untersuchungen sowie die erzielten Ergebnisse werden in Kapitel 10 ausführlich diskutiert.

8.5.1.2 Wissensmodell

Zur Erfüllung der Forderung nach einem differenzierten Angebot an Unterstützungsstrategien (vgl. Soller & Lesgold, 1999, S.4) kommen mehrere Konzepte zur Bereitstellung von Expertenwissen innerhalb des Wissensmodells zur Anwendung. So spielt das Wissen um Lösungen für Probleme oder Problemsituationen, die während der gemeinsamen Programmierung auftreten können, eine wichtige Rolle. Die Umsetzung des eigentlichen Expertenwissen zur Lösung von Problemen kann unter anderem mittels einer CBR-Komponente modelliert werden: Eine Fallbasis enthält typische Problem-Lösung-Tupel, so dass ein Klassifizierer auf Anfrage zu einem bestimmten Problem eine oder mehrere Lösungsvorschläge auswählt und an das anfragende Tutormodell weiterreicht. Ein Lösungsvorschlag kann dabei ein Quelltext sein, der ein bestimmtes Konzept der objektorientierten Java-Programmierung umsetzt und illustriert (vgl. Kölle, 2007).

Aufgrund der zu berücksichtigenden Rollenaspekte findet sich eine zusätzliche Komponente innerhalb des Wissensmodells, die eine echte Erweiterung des Wissensmodells im Vergleich zu typischen *Single-User-ITS* darstellt, also zu solchen ITS, die sich auf die Unterstützung eines einzelnen Lernenden beschränken: Die Integration eines Rollenmodells ermöglicht die angemessene Unterstützung von Lerngruppen, indem die jeweilige Zusammensetzung der Lerngruppe hinsichtlich des verwendeten Rollenmodells bei der Generierung von Problemlösungsangeboten berücksichtigt werden kann. Dazu muss die Rollenkomponente des Wissensmodells geeignete Annahmen bezüglicher der idealen Zusammensetzung einer Lerngruppe als Referenzmodell treffen. Entsprechende Untersuchungen dazu werden derzeit vorbereitet (vgl. Schill, 2007), verwertbare Erkenntnisse sind jedoch noch nicht verfügbar. Das verwendete Rollenmodell wird in Kapitel 9 eingehend diskutiert.

8.5.1.3 Tutormodell

Anhand des Wissensstandes der Lernenden, das heißt der gesamten Lerngruppe, repräsentiert durch Rollenprofile, schließt das Tutormodell durch Vergleich mit dem Referenzmodell des Wissensmodell auf Defizite hinsichtlich vom Team wahrzunehmender Rollen[8]. Entsprechend der erkannten Defizite erfolgt eine Anpassung des Hilfsangebots durch Bewertung und Auswahl der verfügbaren Strategien. Somit nimmt das Tutormodell für den Unterstützungsprozess eine zentrale Rolle ein, indem es die didaktische Strategie auswählt und bereitstellt, anhand derer Unterstützung geleistet wird.

Als weitere Einflussvariable geht die jeweilige Problemsituation in diesen Entscheidungsprozess mit ein. Aus diesem Grunde ist eine Komponente zur Identifizierung von Problemsituationen ein wichtiger Bestandteil des zu erstellenden Tutormodells. Dabei müssen neben rein fachlichen Problemen auch Probleme erkannt werden, die sich aus dem zwischenmenschlichen Miteinander ergeben (s. Kap.7.3.1.3). Analog zur Bestimmung von Rollenprofilen wird auch das Erkennen von aktuellen Problemen auf Basis der Interaktionen der Lernenden mit dem VitaminL-System durchgeführt. Konzepte, Ansätze und konkrete Ergebnisse zu dieser Thematik werden in Kapitel 11 diskutiert.

Ein Kommunikationsmodell als weitere Komponente innerhalb des Tutormodells wird für die Realisierung von Hilfsangeboten in Form von Kommunikationsbeiträgen verwendet: Anstatt eine komplette Problemlösung anhand eines Beispiels in Form eines Quelltextdokuments zu illustrieren, kann beispielsweise bei einem einfachen Systaxfehler in Form eines fehlenden Semikolons ein Hinweis wie *„Ich denke, in Zeile x fehlt ein Semikolon."* effektiver sein. Unterstützungsmaßnahmen, die auf Kommunikation beruhen, können mittels einer Chatterbot-Komponente wie *JAIMBot* (s. Seite 130) realisiert werden.

Bei entsprechender Konfiguration solch einer Kommunikationskomponente verwendet auch diese aussschließlich die Kommunikationselemente der strukturierten

[8] Rolle im Sinne von Teamfunktion

Kommunikation nach dem *CLS*-Ansatz (s. Kap.8.2), so dass mögliche Unterschiede zu den menschlichen Sitzungsteilnehmern im Idealfall nicht mehr wahrnehmbar sind: Der virtuelle Tutor äußert sich wie sein menschliches Gegenstück und wird daher auch als solcher warhgenommen und nicht als eine künstliche Komponente epmfunden, wodurch eine ansonsten mögliche Reduzierung von Glaubwürdigkeit und Akzeptanz seitens der Lernenden vermieden werden kann (vgl. MATESSA, 2001).

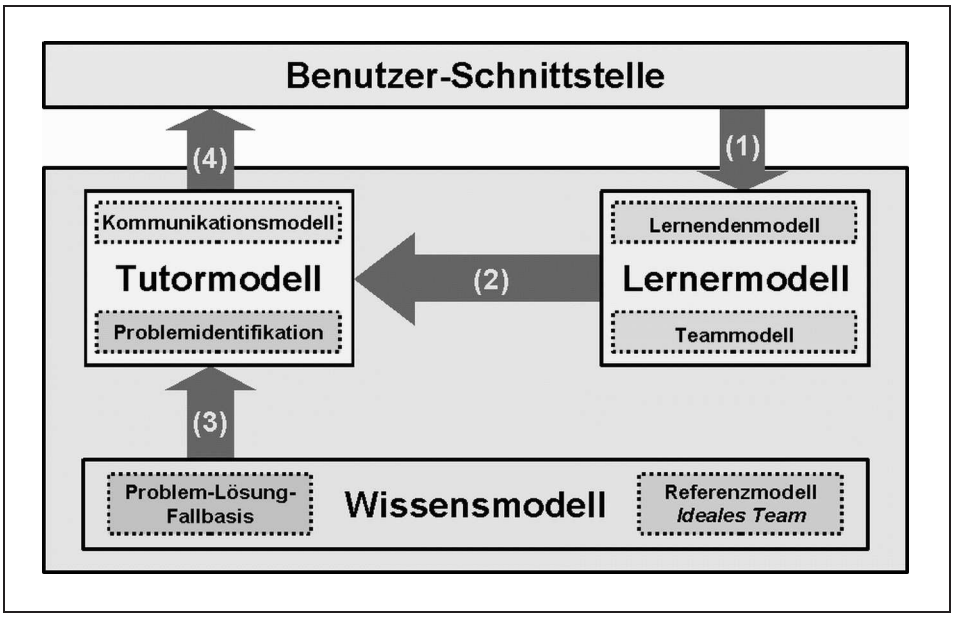

Abb. 8.29: ITS-Architektur innerhalb des VitaminL-Systems

Die Architektur der VitaminL-Tutorkomponente wird in Abbildung 8.29 schematischen zusammengefasst. Dabei sind die in das bereits bestehende CSCL-System zu intergrierenden ITS-Komponenten, namentlich das Lernermodell, das Wissensmodell und das Tutormodell, von der bereits vorhanden Benutzerschnittstelle deutlich abgesetzt. Der geplante Ablauf entspricht dabei weitgehend der in Kapitel 6.3 skizzierten Vorgehensweise.

8.5.2 Adaption der VitaminL-Architektur

Für eine Umsetzung der eben skizzierten Komponente des Tutorsystems ist die bestehende Architektur des VitaminL-Systems (s. Kap.8.4.5) geeignet zu erweitern. Die Schnittstelle, an welcher die Interaktionen des Benutzers mit dem System zur weiteren Analyse abgegriffen werden, ist nicht notwendigerweise die graphische Benutzerschnittstelle (GUI; *Graphical User Interface*): Interpretiert man die laufende Client-Instanz eines Benutzers als dessen Repräsentation innerhalb des VitaminL-Systems, so kann – unter Berücksichtigung der nachrichtenbasierten Client-Server-Kommunikation (s. Kap.8.4.2) – die geforderte Schnittstelle auch zwischen Client und Server positioniert werden. Die Tutorkomponente knüpft in diesem Fall an die server-seitige Verarbeitung empfangener Nachrichtenobjekte an, so dass deren

Weiterverarbeitung – analog zu deren Protokollierung (s. Kap.8.4.4) – innerhalb der Tutorkomponente erfolgen kann. Dazu ist das in Beispiel 8.2 dargestellte Code-Fragment nur geringfügig zu erweitern.

Beispiel 8.8: Verarbeitung empfangener Nachrichtenobjekte auf dem Server

```
// Uebergib das Nachrichtenobjekt msg an den Protokollmechanismus log
log.log(msg);

// Uebergib das Nachrichtenobjekt msg an die Tutorkomponente tutor
// zur weiteren Verarbeitung (d.h. Analyse von Rollen etc.)
tutor.analyze(msg);

// Packe die Nachricht msg zusammen mit ihrem Verarbeiter this
// in einen Nachrichtencontainer evt vom Typ VMessageEvent
VMessageEvent evt = new VMessageEvent(msg, this);

// Fuege den Nachrichtencontainer evt
// an das Ende der Warteschlange dispatcher an
dispatcher.send(new VMessageEvent(msg, this));

// Fertig - bis zur naechsten eintreffenden Nachricht
```

■

Innerhalb der Tutorkomponente wird jede zu analysierende Nachricht sowohl durch das Lernermodell als auch durch das Tutormodell weiterverarbeitet:

- Im Lernermodell erfolgt die Durchführung einer Rollenanalyse für jedes einzelne Teammitglied auf der Grundlage aller von ihm ausgelösten Aktionen. Pro Teilnehmer wird ein Lernendenmodell mit einem enthaltenen Rollenprofil generiert. Die Lernendenmodelle werden durch während einer Benutzersitzung eingehende Nachrichten ständig aktualisiert und ihrerseits zu einem Teammodell zusammengefasst, das für den Lernprozess relevante Informationen über die Zusammensetzung des virtuellen Teams hinsichtlich des verwendeten Rollenmodell enthält. Die im Lernermodell gewonnenen Informationen werden an das Tutormodell weitergereicht.

- Ebenfalls auf der Basis der Benutzeraktionen wird im Tutormodell die Erkennung von Problemsituationen durchgeführt. Für erkannte Probleme kann der virtueller Tutor aktiv werden und unter Berücksichtigung weiterer Informationen wie des Teammodells mit Hilfe des Wissensmodells geeignete Unterstützungsmaßnahmen einleiten. Der virtuelle Tutor nimmt dabei die Funktionen solcher Rollen wahr, die gemäß Teammodell unterbesetzt sind, und kompensiert somit etwaige Rollendefizite, so dass er als eine Art virtuelles Teammitglied aufgefasst werden kann, welches das Team bei Bedarf um fehlende Rollen durch Wahrnehmung der diesen Rollen zugeordneten Teamfunktionen ergänzt.

Abbildung 8.30 fasst die Gesamtarchitektur schematisch zusammen.

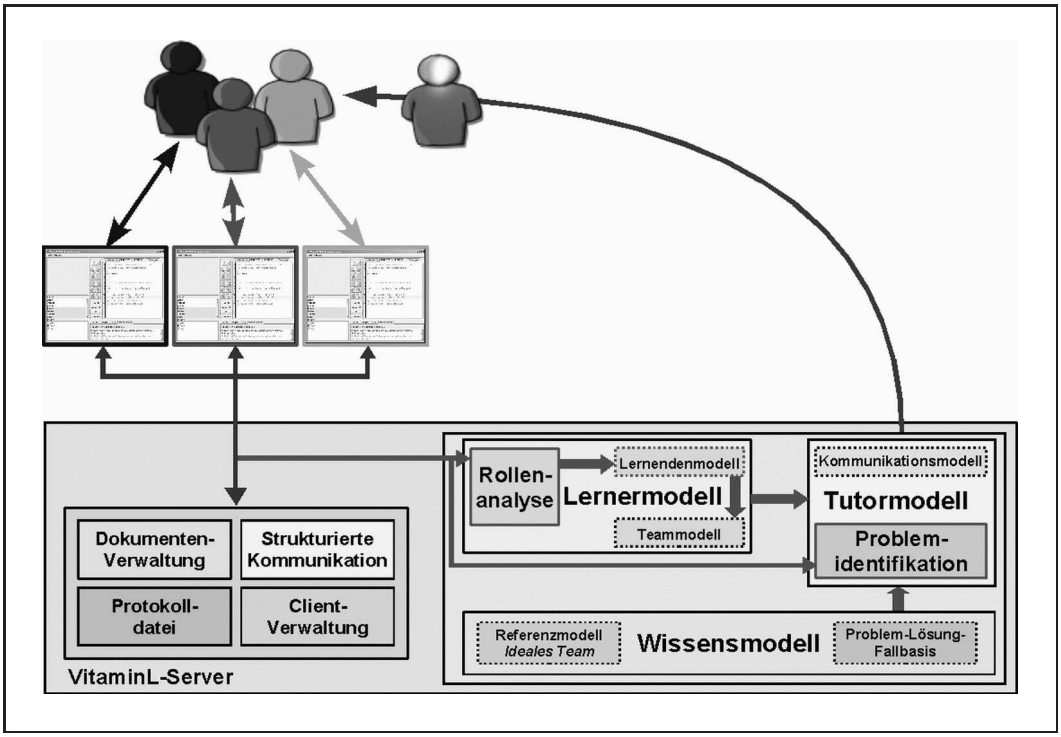

Abb. 8.30: Gesamtarchitektur des VitaminL-Systems als ITS

Links oben in der Abbildung 8.30 ist symbolisch ein beliebiges virtuelles Team dargestellt, das aus drei Teilnehmern besteht: Die einzelnen Teilnehmer interagieren über die Client-Anwendung innerhalb einer Sitzung mit dem VitaminL-System, wodurch entsprechende Nachrichtenobjekte ausgelöst und zur weiteren Verarbeitung an den VitaminL-Server geschickt werden, wo sie in der Sitzungsinstanz der jeweiligen Teilnehmergruppe respektive innerhalb der dort angesiedelten Komponenten für Dokumentenverwaltung, Kommunikation etc. ausgewertet werden und weitere Nachrichtenobjekte mit Ergebnissen zur Folge haben können.

Hinzu kommt nun die zusätzliche Nachrichtenverarbeitung in der Tutorkomponente: Jede auf dem Server eingehende Nachricht wird an die Tutorkomponente weitergereicht und dort wie zuvor beschrieben im Zuge der Rollen- und Problemanalyse ausgewertet. Ergebnisse werden innerhalb der Tutorkomponente weitergereicht, so dass im Bedarfsfall, das heißt bei Erkennen einer akuten Problemsituation, innerhalb des Tutormodells das passende Unterstützungsangebot ausgewählt und den Mitgliedern des virtuellen Teams kommuniziert werden kann.Da hierbei versucht wird, fehlende Rollen einzunehmen, kann der virtuelle Tutor als zusätzliches, künstliches Teammitglied verstanden werden.

Die Entwicklung der Analysekomponente als ein Bestandteil des virtuellen Tutors und der daraus folgende Ausbau des vorliegenden CSCL-Systems zum geplanten Tutorsystem werden in den nun folgenden Kapiteln ausführlich behandelt. Mit dem Ziel, virtuellen Teams in erkannten Problemsituationen tutorielle Hilfestellung unter Berücksichtigung der Teamzusammensetzung anzubieten, liefert die intensive Auseinandersetzung mit den in Kapitel 7.3.4 formulierten Fragestellungen wichtige Erkenntnisse.

Kapitel 9

Das VitaminL-Arbeitsmodell

Das dem in Kapitel 7.1.2 skizzierten Lernprozess zugrundeliegende Arbeitsmodell von VitaminL besteht aus zwei Komponenten, einem Vorgehensmodell und einem Rollenmodell:

- Das Vorgehensmodell von VitaminL beschreibt die prinzipielle Vorgehensweise der an dem Lernprozess Beteiligten und ist in einander ergänzende Prozesse zerlegbar, die ihrerseits aus mehreren kleineren Teilschritten oder Phasen bestehen.
- Das Rollenmodell von VitaminL beschreibt Rollen im Sinne von Teamfunktionen, die von den Teilnehmern während des Lernprozesses im Hinblick auf eine erfolgreiche Durchführung wahrzunehmen sind.

9.1 Das Vorgehensmodell von VitaminL

So unterschiedlich wie die Teilnehmer der Benutzertests sind, so unterschiedlich zeigt sich auch deren Herangehen an die gestellten Programmieraufgaben. Während die Mitglieder einer Gruppe gleich zu Beginn einer Sitzung neue Dokumente erzeugen und anfangen, ohne weitere Absprachen Quelltexte zu verfassen, beginnt in einer anderen Gruppe zunächst eine lebhafte Diskussion, die neben Beiträgen zur Aufgabenstellung auch Diskussionfäden zu Alltagsthemen (sog. *off-topics*) enthalten kann. Im weiteren Verlauf solcher Sitzungen folgt in einigen Gruppen eine Festlegung und Verteilung von Teilaufgaben, wohingegen sich andere Gruppen darauf einigen, dass ein Teammitglied mit der Programmierung beschäftigt ist und die anderen Mitgliedern – ähnlich wie in der Testphase des VitaminL-Prototypen (s. Kap.8.1) – als eher passive Beobachter der Sitzung beiwohnen.

Insgesamt kann aufgrund der Beobachtungen festgestellt werden, dass sich die Vorgehensweise der Gruppen bei der Java-Programmierung überwiegend unstrukturiert darstellt und die Entwicklung der Aufgabenlösungen eher spontan als geplant durchgeführt wird. Folglich ist das Vorgehensmodell von VitaminL auch

nicht als eine Vorgehensweise zu verstehen, die bei der Aufgabenbearbeitung von den Teilnehmern zwingend zu beachten ist, sondern sie stellt vielmehr eine Art empirischen Rahmen dar, in welchem sich die einzelnen Teams während der Java-Programmierung üblicherweise bewegen. Dieser Rahmen beinhaltet einerseits einen fachspezifischen oder fachlichen Prozess, der die eigentliche Aufgabenbearbeitung beinhaltet, und zum anderen einen sozialen Prozess, der parallel zu den Aktivitäten des fachlichen Prozesses stattfindet und sowohl durch die Teilnehmer verursachte Störungen dieses Prozesses als auch Gegenmaßnahmen zur Beseitigung eben dieser Störungen umfasst.

9.1.1 Fachlicher Prozess

Der fachliche Prozess beinhaltet sämtliche fachspizifischen Aktivitäten, die im Kern den der Aufgabenbearbeitung zugrundeliegenden Arbeitsprozess betreffen und die damit direkt zur Aufgabenbearbeitung beitragen. Dieser gesamte Arbeitsprozess lässt sich in zwei aufeinanderfolgende Teilprozesse, die Arbeitsvorbereitung und die Umsetzung, gliedern. Jeder dieser Teilprozesse besteht aus mehreren Phasen, die sich ihrerseits in einzelne Schritte zerlegen lassen. Es wird – vereinfacht – angenommen, dass der fachliche Prozess sequentiell abgearbeitet wird, das heißt beginnend mit dem ersten Teilprozess werden dessen Phasen (und die darin enthaltenen Schritte) nacheinander durchgeführt, so dass dieser Teilprozess abgeschlossen wird, bevor der nächste Teilprozess begonnen wird.

9.1.1.1 Vorbereitung

Die Vorbereitung als der der eigentlichen Umsetzung vorgelagerte Teilprozess dient dem Zweck, die nachfolgende Umsetzung so weit zu organisieren, dass die zu bearbeitende Aufgabe im Prinzip als gelöst betrachtet werden kann. Dies bedeutet, dass enthaltene Probleme erkannt und entsprechenden Lerninhalten zugeordnet werden konnten. Die zugehörigen, in Frage kommenden Lösungsansätze werden als Teilaufgaben unter den Teammitgliedern verteilt. Der Teilprozess der Vorbereitung gliedert sich in Phasen zur Problemerkennung, zur Herstellung des Lernkontexts und zur Organisation der Problemlösung.

9.1.1.1.1 Problemidentifikation Eine Sitzung beginnt üblicherweise mit dem Lesen der Aufgabenstellung, die vor Beginn allen Teilnehmern zugänglich gemacht wird, gefolgt von einer in der Regel kurzen Diskussion, mit welcher das Ziel verfolgt werden sollte, innerhalb einer Gruppe einen Konsens hinsichtlich der Aufgabenstellung zu erzielen. Nur wenn dies allen Mitgliedern klar ist, können anschließend die in der Aufgabe enthaltenen Teilprobleme identifiziert werden.

9.1.1.1.2 Herstellung des Lernkontexts Das Ziel dieser Phase besteht darin, den Bezug zwischen der Aufgabenstellung und den zugehörigen Lerninhalten

herzustellen, indem die in der vorigen Phase identifizierten Teilprobleme bekannten Lerninhalten zugeordnet werden. Anhand der Lerninhalte sind potentielle Lösungsansätze zu ermitteln und im Hinblick sowohl auf die Aufgabenstellung als auch auf die daraus resultierenden Anforderungen zu bewerten. Den Abschluss dieser Phase bildet die Auswahl derjenigen Lösungsansätze, die den Mitgliedern geeignet erscheinen, als Grundlage für die nachfolgende Umsetzung zu dienen.

9.1.1.1.3 Organisation der Problemlösung Die Ergebnisse der vorangegangenen Phase werden nun so aufbereitet, dass im Anschluss nahtlos mit der eigentlichen Umsetzung begonnen werden kann. Dies bedeutet konkret, dass – basierend auf den als geeignet bewerteten Lösungsansätzen – konkrete Teilaufgaben definiert werden. Diese Tätigkeit umfasst auch die Festlegung von Schnittstellen zwischen den Einzelaufgaben, um ein möglichst reibungsloses Zusammenspiel der Einzellösungen zu gewährleisten. Sobald alle Teilaufgaben geeignet auf die Teammitglieder verteilt sind, gilt diese Phase – und damit auch die Vorbereitung als Teilprozess des fachlichen Prozesses – als abgeschlossen.

9.1.1.2 Umsetzung

Mit dem Abschluss der Vorbereitungen erfolgt der Eintritt in die Umsetzung, in welcher die eigentliche Programmierung stattfindet. Das in diesem Teilprozess beobachtete Verhalten der Teilnehmer entspricht im Kern dem des sogenannten *Code-and-fix-Zyklus* (auch als *Build and fix* bekannt; vgl. Zuser et al., 2004, S.70), bei dem zunächst ein Quellcode verfasst wird, der anschliessend solange übersetzt, ausgeführt und überarbeitet wird, bis das Ergebnis den Anforderungen entspricht. Die nachfolgende Abbildung 9.1 stellt diese Vorgehensweise schematisch dar.

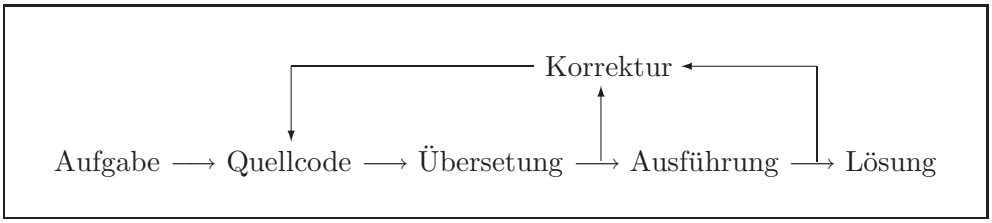

Abb. 9.1: Vorgehensweise nach dem Code-and-fix-Zyklus

Eine etwas differenziertere Betrachtung dieser Vorgehensweise im Zusammenhang mit den durchgeführten Benutzertests lässt eine Strukturierung dieses Vorgangs in mehrere Phasen zu, die im Folgenden näher erläutert werden sollen.

9.1.1.2.1 Codierung der Rohfassung Entsprechend der zuvor definierten und delegierten Teilaufgaben entsteht in der ersten Phase der Umsetzung eine Rohfassung. Üblicherweise suchen die Teilnehmer zunächst passende Templates, das heißt Vorlagen in Form von Code-Beispielen. Dazu greifen die Teilnehmer auf bereits bekannte Beispiele aus Büchern, aus dem Internet und aus dem Skript zurück.

Dies konnte nicht nur in Benutzertests beobachtet werden, sondern wurde auch von den Teilnehmern einer Umfrage zu bevorzugten Informationsquellen bestätigt (s. Kap.7.3.2.1). Die Templates werden anschließend so angepasst, dass sie – zumindest aus der Sicht der Teammitglieder – einer algorithmischen Lösung des zu bearbeitenden Teilproblems entsprechen bzw. nahekommen.

Die einzelnen, auf diese Weise erstellten Code-Stücke werden in einem nächsten Schritt von den Teammitgliedern zu einer ersten Gesamtlösung zusammengesetzt. Diese Gesamtlösung ist naturgemäß noch fehlerbehaftet, das heißt die Übersetzung mit dem Java-Compiler und/oder die Ausführung per Java-Interpreter erzielen noch nicht das gewünschte Ergebnis.

9.1.1.2.2 Fehlerbereinigung Der Weg von der ersten Rohfassung bis zur fertigen Lösung ist – nicht nur in den Benutzertests – durch eine umfangreiche Fehlerbereinigungsphase gekennzeichnet. In dieser Phase müssen zunächst diejenigen Fehler beseitigt werden, die durch den Compiler erkannt und in Form von Fehlermeldungen während des Übersetzungsvorgangs aufgezeigt werden. Der Anwender muss jeden Fehler als solchen identifizieren und lokalisieren können, das heißt er muss in der Lage sein, zum einen den Grund für das Auftreten eines Fehler zu erkennen und andererseits die entsprechende Stelle im Quelltext zu ermitteln. Gerade dies jedoch bereit den Anfängern oftmals Probleme, so dass Unterstützung bei der Fehlererkennung angeraten wird (s. Kap.7.3.1.1).

Zur Behebung eines jeden identifizierten Fehlers ist eine geeignete Maßnahme zu finden und anzuwenden. Dieser Vorgang wird solange wiederholt, bis sämtliche vom Compiler erkannten Fehler behoben sind und die Gesamtlösung frei von syntaktischen Fehlern ist. Erst dann kann eine Ausführung per Interpreter in Betracht gezogen werden. Bei der Ausführung können weitere Fehler auftreten: Laufzeitfehler werden durch das Auftreten sogenannter *Exceptions* (dt.: Ausnahmen) gekennzeichnet. Auch diese Fehler sind – analog zu den Compiler-Fehlern – zu identifizieren, zu lokalisieren und durch Anwendung geeigneter Maßnahmen zu beseitigen.

9.1.1.2.3 Abschlussarbeiten In dieser letzten Phase der Umsetzung werden die (fehlerfreien) Quelltexte als Ergebnis der vorangegangenen Aktivitäten gemäß vorab vereinbarter Regeln und Richtlinien formatiert und mit Kommentaren versehen. Dies kann zwar auch bereits während der Codierung (und der Fehlerbereinigung) geschehen, Beobachtungen haben jedoch gezeigt, dass diese Arbeiten von den Teilnehmern erst dann durchgeführt werden, wenn der Quellcode den Anforderungen der Aufgabenstellung genügt.

Die Formatierung eines Quelltexts dient dem Zweck, diesen in eine einheitliche äußere Form zu versetzen und durch zusätzliche Leerzeilen und Leerzeichen die Übersichtlichkeit und Lesbarkeit zu erhöhen. Dazu gehört auch, mittels Einrückungen die im Quelltext verwendeten Kontrollstruktur hervorzuheben. Das Einfügen von Kommentaren in einen Quelltext trägt überdies zur Verständlichkeit des Quelltexts bei: Mittels zusätzlicher textueller Beschreibungen können gezielt wichtige

Elemente eines Quelltexts (wie Klassen, Attribute, Operationen und Einzelanweisungen) und deren Bedeutung für den Quelltext wie auch für das gesamte Programm beschrieben werden. Diese Beschreibungen können auch die Arbeitsweise sowie die Verwendungsmöglichkeiten von Quelltexten bzw. Programmteilen dokumentieren.

Erst mit Abschluss dieser Phase wird ein Java-Programm als vollständig erachtet.

9.1.1.3 Unterstützung

Parallel zu den beiden soeben charakterisierten Teilprozessen existiert innerhalb des fachlichen Prozesses ein dritter Teilprozess zur Unterstützung der beiden erstgenannten Teilprozesse. Die in diesem Teilprozess enthaltenen Phasen und Schritte begleiten die anderen Aktivitäten des fachlichen Prozesses, können in ihrem zeitlichen Ablauf aber wesentlich stärker variieren als die Aktivitäten der Arbeitsvorbereitung oder der Umsetzung.

9.1.1.3.1 Koordination Die Koordination als Gruppenprozess zur Abstimmung aufgabenbezogener Tätigkeiten (s. Kap.2.2.2) nimmt in einer 90-minütigen Sitzung einen nur geringen Anteil ein. Trotzdem konnten in den absolvierten Benutzertests wiederholt wichtige koordinierende Elemente beobachtet werden. Diese reichen von der Einhaltung des gegebenen Zeitrahmens über die Koordination von Teilaufgaben bis hin zur Überprüfung von erreichten (Teil-)Ergebnissen.

9.1.1.3.2 Informationsbeschaffung Für die Aufgabenbearbeitung wie für den umgebenden Lernprozess gleichsam unverzichtbar ist die Beschaffung von nicht in der Gruppe vorhandenen Informationen aus externen Informationsquellen. Diese Aktivität wird hauptsächlich in Problemsituationen durchgeführt, in denen die Gruppe für eine weitere Aufgabenbearbeitung auf externe Information angewiesen ist. Die Informationsbeschaffung stellt für den gesamten Prozess der Aufgabenbearbeitung eine zentrale Funktion dar, die sich in den Benutzertests in zwei Ausprägungen äußert: Zum einen umfasst die Informationsbeschaffung die selbständige Recherche von Gruppenmitgliedern, bei der auf Literatur, Skripte, im Internet verfügbare Online-Quellen etc. zugegriffen wird, zum andern gehört auch die Möglichkeit, bei Bedarf einen (menschlichen) Tutor zu Rate zu ziehen, zu den typischen Varianten der Informationsbeschaffung, die während der Benutzertests, aber auch bei den in Kapitel 7.1.2 beobachteten Praxiseinheiten beobachtet werden konnten.

Speziell die letztgenannte Form der Informationsbeschaffung bietet einen idealen Ansatzpunkt für den Einsatz eines virtuellen Tutors, der anstelle eines menschlichen Tutors befragt werden kann und sich – im Gegensatz zu seinem menschlichen Pendant – durch eine hohe Verfügbarkeit auszeichnet.

9.1.1.3.3 Fachliche Kommunikation Diese Aktivität umfasst sämtliche Kommunikation innerhalb eines (virtuellen) Teams, die im Rahmen von fachspezifischen Diskussionen zwischen den Teammitgliedern stattfindet und daher unmittelbar der Aufgabenbearbeitung – und somit auch den fachlichen Teilprozessen – zugeordnet werden kann. Als typische Elemente der fachlichen Kommunikation konnten Fragestellungen zur Formulierung von Informationsdefiziten und Erklärungen als Antworten auf Fragestellungen identifiziert werden, die um Meinungen und Ideen ergänzt werden.

Es hat sich bei den durchgeführten Benutzertests gezeigt, dass eine strikte Auftrennung des fachlichen Prozesses in einzelne Teilprozesse, Phasen und Schritte nur rein theoretischer Natur ist und in der Praxis kaum stattfindet. Vielmehr zeichnen sich die in den Benutzertests beobachteten Sitzungen durch folgende Merkmale aus:

- Rückgriffe auf bereits durchgeführte Aktivitäten:
  Obwohl die einzelnen Phasen und Schritte des VitaminL-Vorgehensmodells nicht zwingend in der genannten Reihenfolge auszuführen sind, kann diese doch als eine Art Standard für die Aufgabenbearbeitung in der (virtuellen) Programmiergruppe angesehen werden. In einigen Sitzungen konnte jedoch beobachtet werden, dass einzelne Teammitglieder oder auch die Gruppe als Ganzes zu bereits durchgeführten Aktivitäten zurückkehrten und somit Teile der Aufgabenbearbeitung wiederholten, um beispielsweise bestimmte Sachverhalte neu zu betrachten und bereits getroffene Entscheidungen zu revidieren. In einem Benutzertest schlug beispielsweise einer der Teilnehmer vor, *„...lesen wir doch nochmal die Aufgaenstellung..."*, nachdem 4 Minuten zuvor ein anderes Teammitglied bereits das in der Aufgabenstellung enthaltene Problem benannt hatte: *„Zusammenfassend ganz am schluss soll doch ausgegeben werden, sounsoviel mal ja und soundsoviel mal nein–> benotung ."*.

- Auslassen bzw. Überspringen einzelner Aktivitäten:
  Ein Zeitrahmen von 90 Minuten für die komplette Bearbeitung einer Aufgabe auf der Grundlage des angenommenen Vorgehensmodells bietet für einige der Elemente dieses Modells weniger Zeit als für andere. Die Umsetzung (inkl. aller enthaltenen Aktivitäten) nimmt erfahrungsgemäß den Großteil der zur Verfügung stehenden Zeit in Anspruch, nicht zuletzt auch deshalb, weil die Teilnehmer dies als den Hauptbestandteil des Programmierens und damit einer Programmiersitzung ansehen. Dies kann zur Folge haben, dass andere Aktivitäten des Vorgehensmodells zu kurz kommen oder gar überhaupt nicht durchgeführt werden. In einer Gruppe wurde nach einer kurzen Begrüßung durch den Tutor von einem der Teilnehmer sofort ein Dokument geöffnet und von einem anderen Teammitglied anschließend zu einer Art Aufgabenverteilung übergegangen: *„Würdest Du bitte den ansatz machen, und ich perfektioniere, wie immer...?"*. Wichtige Aktivitäten der Arbeitsvorbereitung wurden in diesem Fall quasi komplett übersprungen und mussten im weiteren Verlauf der Sitzung durch Rücksprung aufgegriffen werden, um aufgetretene Probleme bei der Aufgabenbearbeitung lösen zu können.

- Überschneidungen zwischen aufeinanderfolgende Aktivitäten:
  Es existieren keine trennscharfen Übergänge zwischen Teilprozessen, Phasen und Schritten. So konnte in mehreren Sitzungen beispielsweise beobachtet werden, dass ein Teammitglied bereits mit der Programmierung begonnen hatte, während ein anderes Teammitglied noch mit dem Lesen der Aufgabenstellung beschäftigt war.

Insgesamt muss die in den allermeisten Sitzungen beobachtete Vorgehensweise als weitestgehend unstrukturiert, um nicht zu sagen chaotisch, charakterisiert werden.

9.1.2 Sozialer Prozess

Neben dem rein fachlichen Prozess, der primär zur Aufgabenbearbeitung beiträgt, konnten in den Sitzungen weitere Aktivitäten beobachtet werden, die zunächst als Störung der Aufgabenbearbeitung betrachtet werden müssen. Es wird hier von einem sozialen Prozess – in Abgrenzung zum fachlichen Prozess – gesprochen, da sämtliche diesem Prozess zugeordneten Aktivitäten personenorientiert statt aufgabenorientiert einzustufen sind. Im Mittelpunkt dieses Prozesses stehen also stets die Teammitglieder, die auch als Hauptbeteiligte am sozialen Prozess bezeichnet werden können, der fachliche Prozess und die darin enthaltene Aufgabenbearbeitung bildet lediglich einen Rahmen, innerhalb dessen die Aktivitäten des sozialen Prozesses stattfinden.

Die den verschiedenen sozialen Teilprozessen zugeordneten Störungen sind im Hinblick auf eine Fortsetzung der Aufgabenbearbeitung mit entsprechenden Maßnahmen vom Team und seinen Mitgliedern zu beheben. Die jeweiligen Maßnahmen zur Störungsbehebung werden wie die Störung selber als Bestandteil des sozialen Prozesses bzw. einer dem sozialen Prozess zugeordneten Aktivität betrachtet, die mit erfolgreicher Anwendung einer geeigneten Maßnahme endet. In den nun folgenden Kapiteln werden diejenigen sozialen Aktivitäten skizziert, die während der durchgeführten Benutzertests aufgetreten sind, wobei im Unterschied zum fachlichen Prozess keine weitere Verfeinerung in Teilphasen oder Schritte unternommen wird.

9.1.2.1 Off-Topic-Aktivitäten

Zu den am häufigsten beobachtete Störungen zählen die sogenannten *off-topic-Aktivitäten*. Diese äußern sich während der Zusammenarbeit im VitaminL-System durch Kommunikationsbeiträge zu solchen Themen, die weder inhaltlich der zu bearbeitenden Aufgabe zugeordnet werden können noch einen konstruktiven Beitrag im Hinblick auf eine erfolgreiche Problemlösung leisten.

So wurden in einer Sitzung spontane Beiträge wie *„Lasst es mich so erklären: Alf ist cool."* oder auch *„Weißt Du, der Kuchen ist lecker."* formuliert, auf die die anderen Mitglieder allerdings nicht weiter eingingen, so dass diese Äußerung keine

länger dauernde Störung auslöste, sondern anschließend schon wieder als beendet galt.

In einer anderen Sitzung machte sich anscheinend eines der Mitglieder mit dem Bedienkonzept der Dialogschnittstelle vertraut, indem es eine Reihe zusammenhangloser Kommunikationsbeiträge auslöste, beginnend mit *„Sehr gut."* über *„Okay, lasst uns fortfahren."* bis *„Das ist richtig"*. Daraufhin griff ein anderes Teammitglied ein und ermahnte den Störungsverursacher, diese Vorgehensweise zu beenden (*„Ich bin ziemlich sicher: 'Sei jetzt ruhig'."*). Anschließend wandten sich alle Teammitglieder wieder ihren eigentlichen Aufgaben zu (hier: der Bearbeitung von Quelltexten).

Eine weitere Sitzung wurde von einem Mitglied mit dem Beitrag *„Ich denke, wir sollten erstmal Kaffee trinken."* eröffnet, woraufhin ein anderes Teammitglied zunächst Zustimmung signalisierte (*„Sehr gut."*), anschließend aber zum Weiterarbeiten aufforderte (*„Okay, lasst uns fortfahren."*) und auf die Notwendigkeit der Aufgabenverteilung hinwies (*„...können wir die Aufgaben verteilen, oder wollen wir alle bei null anfangen?"*). Damit war diese Störung beendet und die Aufgabebearbeitung konnte innerhalb der Gruppe fortgesetzt werden.

9.1.2.2 Motivationsprobleme

Unmotivierte Teilnehmer stellen ein weiteres, in den durchgeführten Sitzungen oft beobachtetes Problem dar. Die Gründe für eine fehlende Motivation sind vielfältig. So gibt es zweifelsfrei Teilnehmer, bei denen die Java-Programmierung nur als leidige Pflichtveranstaltung im Rahmen ihres Studiums angesehen wird. Die geringe (extrinsische) Motivation dieser Teilnehmer zeigt sich exemplarisch anhand von Bemerkungen wie *„wollen wir aufhören...?"*, die in dieser oder ähnlicher Form von mehreren Teilnehmern geäußert wurden – in der Regel im letzten Drittel einer Sitzung mit dem offensichtlichen Ziel, diese so schnell wie möglich zu beenden. Noch offensichtlicher formulierte einer der Teilnehmer während einer Sitzung den Beitrag *„ich habe keine Lust mehr..."* .

In Fällen unzureichender Motivation hat es sich als hilfreich gezeigt, wenn ein anderes Teammitglied mit entsprechenden Beiträgen dieses Motivationsdefizit beseitigen kann, so dass ein Weiterarbeiten ermöglicht wird. Dies kann je nach Situation durch Aufzeigen von Lösungsansätzen oder in Form aufmunternder Worte geschehen.

9.1.2.3 Inaktivitäten

Ein weiteres Problem stellen Inaktivitäten einzelner Mitglieder dar, das heißt Zeiträume innerhalb einer Sitzung, in denen ein Teammitglied keine aktiven Beiträge zur Aufgabenbearbeitung leistet. In den durchgeführten Sitzungen konnten verschiedene Umstände beobachtet werden, die zu Inaktivitäten einzelner Mitglieder führten:

1. Mangelhafte Aufgabenverteilung:
   Die Aufgabenverteilung hat entweder gar nicht stattgefunden oder wurde derart ungeschickt durchgeführt, dass einem Teammitglied gar keine Aufgabe oder eine nur ungeeignete Aufgabe zugewiesen wurde.

2. Zurückhaltendes Teammitglied:
   Gerade bei eher introvertierten Teilnehmer konnte oft beobachtet werden, dass diese bei auftretenden Problemen während der Aufgabenbearbeitung eher anfangen, ziellos innerhalb von Dokumenten und zwischen den Dokumenten zu navigieren, anstatt explizit andere Teammitglieder (oder den Tutor) um Rat zu fragen.

3. Demotiviertes Teammitglied:
   Wird die bereits im vorigen Kapitel erläuterte Störung von den anderen Teammitgliedern nicht erkannt (und somit nicht behoben), so kann eine kurze Phase der Demotivation leicht in eine länger dauernde Inaktivität übergehen – der Übergang zwischen diesen Phase ist nahezu fließend.

4. Ungenügende Gruppenarbeit:
   In einigen Situationen hat ein Teammitglied zwar ein fachliches Problem geäußert, aber von seinen anderen Teammitgliedern keinerlei Rückmeldung erhalten, weil diese zum Beispiel in ihre eigenen Aufgaben vertieft und somit nicht aufmerksam genug hinsichtlich der Belange anderer Mitglieder waren. Diese fehlende Unterstützung durch die Gruppe hindert das Mitglied nicht nur an der Fortsetzung seiner Arbeit, sondern kann es auch so weit demotivieren, dass es in Inaktivität verfällt.

Je nach Ursache einer Inaktivität existieren diverse Maßnahmen, solch eine Störung zugunsten der Fortführung des fachlichen Prozesses zu beenden. So kann unter anderem eine Neuverteilung der Aufgaben erfolgen, um nicht-aktive Mitglieder wieder an der Gruppenarbeit zu beteiligen. Das Erkennen von Fachproblemen anderer Teammitglieder und die Initiierung einer klärenden fachlichen Diskussion kann in anderen Fällen als geeignete Maßnahme zur Anwendung kommen.

9.1.2.4 Konflikte

Als Konflikt wird in dieser Arbeit eine Störung der Teamarbeit aufgefasst, die sich aus unterschiedlichen Ansichten beziehungsweise Meinungen der Teilnehmer dahingehend entwickelt, dass die Kommunikation den fachlichen Prozess weitgehend verlässt und in zum Teil beleidigenden Äußerungen endet. Während der gesamten Projektdauer konnte lediglich bei einer Gruppe solch ein Verhalten beobachtet werden, das sich darüber hinaus auf mehrere Benutzertests erstreckte. Aus Gründen der besseren Lesbarkeit sind die entsprechenden Sitzungsausschnitte, die die entsprechenden Kommunikationsbeiträge enthalten, mitsamt Erläuterungen im Anhang (Kap.E.4) wiedergegeben.

Als Folge der verzeichneten Konflikte konnte eine nicht unerhebliche Störung der Aufgabenbearbeitung beobachtet werden, die letztlich nur dadurch behoben werden konnte, indem ein an dem Konflikt unbeteiligtes Mitglied durch seine Verhaltensweise wieder den Fokus auf die Aufgabenstellung lenkte. Eine Moderation durch eine möglichst neutrale Instanz kann somit in Konflikten eine mögliche Unterstützungsstrategie darstellen.

9.1.2.5 Kommunikationsprobleme

Eine Vielzahl der beobachteten sozialen Aktivitäten lässt sich durch Probleme erklären, die ihre Ursachen in der Kommunikation haben. Als typische Erscheinungsformen können beispielsweise Überschneidungen mehrerer Themenstränge genannt werden (vgl. Göldner, 2005, S.56f). Auch Redundanzen lassen sich oft beobachten: Ein Teammitglied liefert einen Kommunikationsbeitrag, der jedoch keine Beachtung durch die anderen Teammitglieder findet, so dass aufgrund dieser fehlenden Rückmeldung eine Wiederholung stattfindet (vgl. Göldner, 2005, S.58f).

Auch der Chat selbst als Werkzeug der Kommunikation kann Ursache für einige der beobachteten Probleme sein: Beim Verfassen eines längeren Beitrags hat es sich als durchaus üblich und praktikabel erwiesen, diesen in kleinere Teile zu zerlegen, um Wartezeiten auf der Empfängerseite so gering wie möglich zu halten. Dabei kann es unter Umständen sogar zur Überschneidung zweier Beiträge kommen (vgl. Göldner, 2005, S.76f).

Auf eine umfassende sprachliche Untersuchung sämtlicher Kommunikationsprobleme muss in der vorliegendenArbeit verzichtet werden, jedoch sei an dieser Stelle auf die Arbeit von Göldner (2005) hingewiesen, die sich eingehend mit Problemsituationen in der Chat-Kommunikation im Rahmen objektorientierter Programmierung befasst.

9.1.3 Zusammenfassung des Vorgehensmodells

Das gesamte Vorgehensmodell von VitaminL ist schematisch in der Abbildung 9.2 wiedergegeben. Ausgehend von der Aufgabenstellung erfolgt die Aufgabenbearbeitung, an deren Ende die fertige Lösung steht. Den empirischen Rahmen für die Bearbeitung der Aufgabe durch das (virtuelle) Team bildet das VitaminL-Vorgehensmodell, das sich durch einen fachlichen Prozess und einen begleitenden sozialen Prozess beschreiben lässt. Der fachliche Prozess umfasst die eigentliche Bearbeitung durch das Team und lässt sich in mehrere Phasen gliedern, zwischen denen Übergänge – symbolisiert durch Pfeile – exisitieren. Der dazu parallel stattfindende soziale Prozess äußert sich in diesem Modell in Form von Störungen des fachlichen Prozesses, die in der schematischen Darstellung durch Pfeile, die vom sozialen Prozess ausgehen und in den fachlichen Prozess hineinragen, symbolisiert werden.

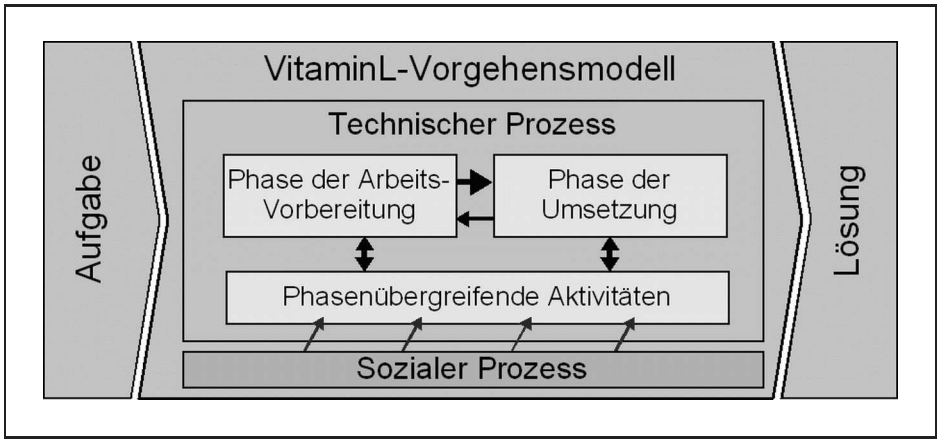

Abb. 9.2: Schematische Darstellung des VitaminL-Vorgehensmodells

Die Teamfunktionen, die für ein erfolgreiches Durchlaufen des Vorgehensmodells erforderlich sind, werden durch das VitaminL-Rollenmodell spezifiziert.

9.2 Das Rollenmodell von VitaminL

Wie bereits in Kapitel 2.5 aufgezeigt wird allgemeinhin davon ausgegangen, dass für eine erfolgreiche Aufgabenbearbeitung mehrere unterschiedliche, sich ergänzende Teamfunktionen benötigt werden. Entsprechend dieser Annahme kommt auch im VitaminL-Projekt ein Rollenmodell zum Einsatz. Dessen Grundlage bildet das von Spencer & Pruss (1995) aufgestellte Rollenmodell (s. Kap.2.5.5), nicht zuletzt wegen dessen universeller Anwendbarkeit: Durch Adaption der einzelnen Rollen an den Lernprozess bzw. das in ihm enthaltene Vorgehensmodell werden die VitaminL-Rollen definiert.

Bei der Ausgestaltung der einzelnen Rollen innerhalb der Tutorkomponente des VitaminL-Systems sind zwei Blickwinkel einzunehmen:

1. Die *externe Operationalisierung* legt fest, wie sich eine Rolle im VitaminL-System gegenüber anderen Teammitgliedern darstellt.

2. Die *interne Operationalisierung* umfasst die Realisierung der jeweiligen externen Operation innerhalb der Tutorkomponente mit geeigneten softwaretechnischen Mitteln und Werkzeugen wie beispielsweise Klassifizierern und Chatbots.

Die Operationalisierung der Rollen soll jedoch an dieser Stelle nicht weiter vertieft werden, da der Schwerpunkt der vorliegenden Arbeit auf der Erkennung von Rollen liegt, während die Operationalisierung hauptsächlich bei der Simulation von Rollen Verwendung findet. Für weiterführende Informationen sei daher an dieser Stelle auf Kölle (2007) verwiesen.

9.2.1 Charakterisierung der Rollen

Ausgehend von den ursprünglichen Rollen gemäß Spencer & Pruss (1995) ergeben sich die in den Kontext der objektorientierten Programmierung eingebetteten Rollen des VitaminL-Projekts. Das jeweilige Pendant einer VitaminL-Rolle im Rollenmodell von Spencer & Pruss (1995) wird in den nachfolgenden Charakterisierungen jeweils in Klammern genannt, zusammen mit Verweisen auf die zugehörigen Beschreibungen. Aus der vom VitaminL-System unterstützten Lernsituation ergeben sich einige Rahmenbedingungen, die weitere Auswirkungen auf die Anwendung der VitaminL-Rollen in einem zukünftigen Tutorensystem zur Folge haben.

9.2.1.1 Die Rolle *Planer*

Der Planer (Visionär; s. Kap.B.3.1.1) zeigt sich für die Gesamtplanung des Projekts verantwortlich und ist weniger an Details interessiert. Infolgedessen ist er an denjenigen Teilprozessen beteiligt, die sich mit der Vorbereitung des Projekts sowie mit der Überprüfung erzielter Ergebnisse befassen. Da die maximale Dauer einer praktischen Lerneinheit – abgesehen vom Abschlussprojekt – auf ein paar Stunden begrenzt ist und somit eine obere Schranke für die unsprünglich langfristige Ausrichtung des Planers darstellt, entfallen langfristige, eher zukunftsorientierte Aspekte dieser Rolle wie beispielsweise die Einbindung des Gruppenziels in andere, übergeordnete (Unternehmens-)Ziele. Somit konzentriert sich der Aufgabenbereich des Planers auf organisatorische Tätigkeiten.

Konkret bedeutet dies eine Teilnahme des Planers an der Problemerkennungsphase und der Verteilung der Aufgaben. An der dazwischen angesiedelten Phase zur Herstellung des Lernkontexts hingegen beteiligt sich der Planer tendentiell nicht, da es dort um ihm lästige Detailarbeit geht. Auch ist der Anteil des Planers an der Umsetzung eher gering, denn er wird sich naturgemäß nicht für die Detailarbeit, sondern nur für das ausführbare Programm (als zu erreichendes Gruppenergebnis) interessieren. In den phasenübergreifenden Teilprozessen bringt sich der Planer in die Koordiniation und in die fachliche Kommunikation ein, die Informationsbeschaffung hingegen überlässt er anderen. Innerhalb des sozialen Prozesses wird der Planer aktiv, wenn er bei anderen Teammitgliedern Off-topic-Aktivitäten feststellt, da er dadurch die Erreichung des Gruppenziels gefährdet sieht. Er wird jedoch kaum selber aktiv, sondern die Ergreifung geeigneter Maßnahmen an den Berater (s. Kap.9.2.1.8) delegieren. Dementsprechend tritt der Planer in dem Tutorsystem von VitaminL zu Zwecken der Zeitplanung und der Organisation der Aufgabenverteilung in Erscheinung (vgl. Kölle & Langemeier, 2005, S.8ff).

9.2.1.2 Die Rolle *Problemlöser*

Der Problemlöser (Pragmatiker; s. Kap.B.3.1.2) als Gegenstück zum Planer ist im Team der Lieferant für Ideen, bei denen auch Randbedingungen beachtet werden,

so dass die Ideen realistisch und umsetzbar sind. Er unterstützt seine Teammitglieder in der Arbeitsvorbereitung einerseits bei der Herstellung des Lernkontexts, indem er mögliche Lösungsansätze in den relevanten Lerninhalten ermittelt, anschließend bewertet und für die Umsetzung auswählt, andererseits konkretisiert der Problemlöser diese soweit, dass sich aus den gewählten Ansätzen Teilaufgaben ableiten lassen.

Während der Umsetzung unterstützt der Problemlöser andere Teammitglieder (insbesondere den Umsetzer; s. Kap.9.2.1.7) bei der Anpassung von Teillösungen und deren Integration in eine Gesamtlösung. Treten in diesem Teilprozess noch Fehler auf, unterstützt der Problemlöser bei deren Lokalisierung sowie bei der Auswahl und Anwendung geeigneter Maßnahmen zu deren Beseitigung. Mit seinen Fähigkeiten bringt er sich darüber hinaus auch in die Phase der Informationsbeschaffung und die fachliche Kommunikation ein, wo er seiner Grundfunktion als Ideenlieferant nachkommt und in entsprechenden Diskussionen Erklärungen liefert und Ideen anbietet.

Am sozialen Prozessen ist der Problemlöser in der Regel nicht beteiligt: Die dort auftretenden Probleme sind nicht fachlicher, sondern sozialer Natur und werden von anderen Rollen beziehungsweise deren Trägern wahrgenommen und behandelt. Die externe Ausgestaltung des Problemlösers erfolgt gemäß Kölle & Langemeier (2005) unter anderem durch die Präsentation von übersetzten Fehlermeldungen (s. Kap.9.1.1.2.2).

9.2.1.3 Die Rolle *Informationsbeschaffer*

Der Informationsbeschaffer (Entdecker; s. Kap.B.3.1.3) kommt stets dann zum Einsatz, wenn das Team bei der Programmierung nicht mehr weiter weiß und Informationen aus externen Quellen (Literatur, Skripte, Internet-Quellen etc.) benötigt. Diese fehlenden Informationen zu beschaffen ist seine Aufgabe. Der Informationsbeschaffer stellt somit eine zentrale Funktion während der gesamten Aufgabenbearbeitung und damit auch für den Lernprozess dar.

Insbesondere in der Umsetzung kann die Tutorkomponente von VitaminL die Rolle der Informationsbeschaffung durch die Präsentation von Beispielen einnehmen (vgl. Kölle & Langemeier, 2005, S.8ff).

9.2.1.4 Die Rolle *Fragesteller*

Der Fragesteller (Herausforderer; s. Kap.B.3.1.4) wird in der Regel dann aktiv, wenn es gilt, Sachverhalte kritisch zu hinterfragen und Entscheidungen bezüglich der weiteren Vorgehensweise zur Aufgabenbearbeitung zu treffen. Insofern ist der Fragesteller auch ein Infragesteller.

Für den fachlichen Prozess bedeutet dies eine Beteiligung des Fragestellers an jeglicher fachlicher Kommunikation: Er versteht es dort, durch Fragestellungen und Meinungsäußerungen die Diskussion aufzurecht zu halten (bzw. im Bedarfsfall auch

anzuregen). In Entscheidungssituationen wie beispielsweise bei der Bewertung von Lösungsansätzen zur Herstellung des Lernkontexts oder auch bei der Auswahl geeigneter Maßnahmenm zur Beseitigung von Übersetzungs- und Laufzeitfehlern ist es immer wieder hilfreich, wenn Sachverhalte kritisch bis provokant hinterfragt werden. Diese Eigenschaft kann der Fragesteller auch im sozialen Prozess für das Team gewinnbringend einsetzen, indem er in Konfliktsituationen und bei Kommunikationsproblemen die Ursachen für die jeweilige Störung hinterfragt. Auch im Falle von Inaktivitäten könnnen durch den Fragesteller losgetretene Provokationen inaktive Mitarbeiter dazu bewegen, ihre Aktivitäten wieder aufzunehmen und die Aufgabenbearbeitung fortzusetzen.

Dementsprechend besteht die externe Operationalisierung des Fragestellers in einem Tutorsystem vornehmlich in der Anregung von Diskussionen im weitesten Sinne.

9.2.1.5 Die Rolle *Moderator*

Der Moderator (Unparteiischer; s. Kap.B.3.1.5) kann in Problemsituationen zum Team hinzugezogen werden, um – ähnlich wie der Fragesteller – bestehende Sachverhalte kritisch zu hinterfragen, das Team neu zu beleben und voranzubringen. Somit ergibt sich auf den ersten Blick zwar eine große Überdeckung zwischen den Aufgabenbereichen des Moderators und des Fragestellers, jedoch kommen den beiden Rollen durchaus unterschiedliche Bedeutungen zu: Während der Fragesteller als festes Teammitglied am gesamten Entwicklungsprozess beteiligt ist, wird der Moderator als eine Art Feuerwehr erst bei auftretenden Krisensituationen hinzugezogen. Demzufolge kommt dieser vornehmlich im sozialen Prozess zum Einsatz, um beispielsweise in Konflikten zu vermitteln oder bei der Klärung von Missverständnissen zu unterstützen. Als von außen hinzugezogenes Teammitglied kann der Moderator auch als Unterstützer des Beraters (s. Kap.9.2.1.8) angesehen werden.

9.2.1.6 Die Rolle *Schlichter*

Der Schlichter (Friedensstifter; s. Kap.B.3.1.6) kommt bei drohenden oder akuten Konfliktsituationen zum Einsatz. Er sieht seine Aufgabe darin, in solchen Situation schlichtend einzugreifen und eine Art harmonisches Gleichgewicht innerhalb des Teams wiederherzustellen. Die Aktivitäten des Schlichters sind daher stets als Teil des sozialen Prozesses zu verstehen: er vermittelt in Konflikten, interveniert bei Off-topic-Aktivitäten und bereinigt Kommunikationsprobleme (wie beispielsweise Missverständnisse) zwischen den Teammitgliedern.

Das Bereinigen sozialer Problemsituationen als externe Operation dieser Rolle beinhaltet ausschließlich geeignete Kommunikationsbeiträge, so dass die interne Umsetzung mittels Chatbot erfolgen kann (vgl. Kölle & Langemeier, 2005, S.8ff).

9.2.1.7 Die Rolle *Umsetzer*

Die Hauptaufgabe des Umsetzers (Arbeitstier; s. Kap.B.3.1.7) besteht darin, Lösungsansätze zu konkretisieren, das heißt er ist hauptsächlich mit der Erstellung der Quelltexte von der ersten Rohfassung bis hin zum ausführlich kommentierten und korrekt formatierten Programmcode, der den Anforderungen der Aufgabenstellung entspricht. Sein Anteil an anderen Phasen des fachlichen Prozesses fällt naturgemäß vernachlässigbar gering aus. Auch an den diversen Aktivitäten des sozialen Prozesses ist der Umsetzer, begründet durch seine starke Aufgabenorientierung, im allgemeinen nicht beteiligt.

Gemäß dem in Kapitel 7.1.3 erläuterten didaktischen Ansatz stellt die praktische Umsetzung theroetischer Inhalte einen wesentlichen Aspekt beim Erlenern einer Programmiersprache dar. Entsprechend hoch ist die Bedeutung des Umsetzers für diesen Prozess und die zugehörige Lernsituation einzustufen. Von einer Unterstützung dieser Rolle durch das VitaminL-System in Form einer Simulation ist unbedingt abzuraten, da der persönliche Lernprozess aller Teilnehmer dadurch um ein wichtiges Element reduziert würde.

9.2.1.8 Die Rolle *Berater*

Die prinzipielle Aufgabe des Beraters (Trainer; s. Kap.B.3.1.8) besteht darin, das Team voranzubringen. Diese Rolle übt der Berater sowohl im fachlichen wie im sozialen Prozess aus. Im Rahmen des sozialen Prozesses zeigt sich der Berater als die treibende Kraft innerhalb des Teams, die – oftmals in Zusammenarbeit mit dem Moderator – wieder motiviert, bei Off-topic-Aktivitäten interveniert und inaktive Teammitglieder zur Fortführung ihrer aufgabenbezogenen Tätigkeiten bewegt.

Im Rahmen des fachlichen Prozesses arbeitet der Berater aufgrund seines Erfahrungshorizonts oftmals eng mit dem Planer zusammen, begibt sich aber – im Unterschied zum Planer – auch auf die Detailebene herunter. Dies zeigt sich im Zusammenspiel mit anderen Rollen wie beispielsweise dem Informationsbeschaffer, dem Problemlöser oder auch dem Umsetzer.

9.2.1.9 Die Rolle *Archivar*

Der Archivar (Bibliothekar; s. Kap.B.3.1.9) ist das Gruppengedächtnis, dessen Aufgaben darin bestehen, einerseits alle während der Zusammenarbeit anfallenden Informationen aufzuzeichnen, um sie andererseits im Bedarfsfall den übrigen Teammitgliedern zur Verfügung zu stellen. Aus der Wahrnehmung der erstgenannten Aufgabe, der vollständigen Protokollierung sämtlicher Gruppenaktivitäten inklusive Kommunikation, folgt prinzipiell die Beteiligung des Archivars an sämtlichen Aktivitäten des fachlichen und des sozialen Prozesses. In seiner zweiten Eigenschaft fungiert der Archivar als Informationslieferant und ist aktiv an denjenigen Aktivitäten beteiligt, die eine Suche nach (bereits vorhandenen) Informationen

beinhalten. Er greift dazu auf seine Aufzeichnungen zurück und ist somit auf bereits vorhandene Informationen bschränkt. Die Suche nach neuen, innerhalb der Gruppe noch nicht bekannten Informationen ist weiterhin Aufgabe des Informationsbeschaffers. Analog zum Planer ist auch beim Archivar der eingeschränkte Zeitrahmen typischer Praxiseinheiten zu berücksichtigen, was zur Folge hat, dass langfristige Aspekte dieser Rolle unberücksichtigt bleiben dürfen.

In der VitaminL-Software ist der Archivar in seiner Rolle als Protokollant bereits vollständig umgesetzt: Die Kommunikationshistorie (s. Kap.8.1.1.2) erlaubt allen Teammitgliedern jederzeit Zugriff auf die gesamte während einer Sitzung geführte Kommunikation. Dieser Mechanismus wird auf einer tieferliegenden technischen Ebene des Systems komplettiert: Der Server protokolliert sämtliche Aktivitäten aller Teammitglieder während einer Sitzung (s. Kap.8.4.1.1.2). Diese Protokolle und die darin enthaltene Information sind den Teilnehmern während einer Situng zwar nicht direkt zugänglich, können aber nach erfolgter Sitzung für spezielle Auswertungen herangezogen werden.

Die Suche und die Präsentation von (vorhandenen) Informationen (in Form von benutzten Dateien und von Kommunikationsbeiträgen) können als externe Operationalisierung des zweiten Aspekts des Archivars ausfgefasst werden.

9.2.1.10 Die Rolle *Vertrauensperson*

Die Vertrauensperson (Beichtvater; s. Kap.B.3.1.10) ist der Ansprechpartner in sämtlichen Problemsituationen, die nicht-fachlicher Natur sind. Sein Engagement übt er folglich im sozialen Prozess aus, vorwiegend bei der Vermittlung in Konflikten und bei Kommunikationsproblemen im weitesten Sinne.

Hierbei ist jedoch die besondere technische Umsetzung der Kommunikation zu berüchsichtigen: Von einem Teammitglied zum Zwecke der Kommunikation verfasste Beiträge werden vom VitaminL-System stets an alle Teammitglieder verteilt – und zwar auch dann, wenn ein Beitrag explizit an ein bestimmtes Teammitglied adressiert ist (sog. dedizierter Empfänger; s. Kap.8.2.4). Darüber hinaus werden in einer Sitzung sämtliche Kommunikationsbeiträge in einer Historie gespeichert, die jedem Teammitglied unmittelbar nach dessen erfolgreicher Anmeldung am System komplett übertragen und somit zugänglich gemacht wird.

Es ist offensichtlich, dass diese Form der Kommunikation keinen Raum für private Dialoge oder sonstige vertraulich zu behandelnde Gespräche bietet. Als Konsequenz daraus ergibt sich eine Entbehrlichkeit des Beichtvaters (resp. der Vertrauensperson), aus welcher sich wiederum der Wegfall der Unterstützung dieser Rolle in der Tutorkomponente des VitaminL-Systems ableiten lässt.

9.2.2 Rollenzugehörigkeit

Die Frage nach dem Grad der Zugehörigkeit eines Teammitglieds zu einer der Rollen ergibt sich konsequenterweises aus der Existenz eines Rollenmodells. In

der Praxis werden als teamdiagnostische Instrumente sowohl Fragebögen (s. Kap.2.4.3.3) als auch Beobachtungen (s. Kap.2.4.3.2) verwendet. Die nachfolgende Tabelle 9.1 gibt einen kurzen Überblick über die in dieser Arbeit skizzierten Rollenmodelle und die jeweils verwendeten teamdiagnostischen Werkzeuge. Ein + in einem Tabellenfeld markiert dabei die bei einem Rollenmodell zum Einsatz kommenden Werkzeuge wie Beobachtung und Befragung.

Tab. 9.1: Rollenmodelle und teamdiagnostische Werkzeuge

Rollenmodell	**Beobachtung**	**Befragung**
Bales & Slater (2.5.1)	+	
Belbin (2.5.2)	+	+
Margerison & McCann (2.5.3)		+
Eunson (2.5.4)	+	
Spencer & Pruss (2.5.5)		+

Da psychologische Tests (s. Kap.2.4.3.1) bei keinem der genannten Rollenmodelle zum Einsatz kommen, werden diese in der Tabelle 9.1 nicht aufgeführt.

9.2.2.1 Der Fragebogen von Spencer & Pruss

Spencer & Pruss haben einen 150 Aussagen starken Fragebogen entwickelt, mit dessen Hilfe die Ausprägungen eines Teilnehmers hinsichtlich der zehn Rollen des Rollenmodells von Spencer & Pruss bestimmt werden. Der Fragebogen enthält pro Rolle 15 Aussagen, die jeweils mit drei Antwortmöglichkeiten versehen sind (s. Tab.9.2). Der komplette Fragebogen ist bei Bedarf im Anhang, Kapitel B.3.2 einsehbar.

Tab. 9.2: Antwortoptionen des Teamfragebogens von Spencer & Pruss

Antwort	**Bedeutung**
1	*„Da stimme ich überhaupt nicht überein"* oder *„Das trifft auf mich eigentlich nicht zu"*
2	Weder völlige Zustimmung noch völlige Ablehnung
3	*„Da stimme ich voll zu"* oder *„Das trifft genau auf mich zu"*

(vgl. Spencer & Pruss, 1995, S.75)

Wie der Tabelle zu entnehmen ist, wird jede Antwort mit der ihr zugeordneten Punktzahl von 1 bis 3 Punkten gewichtet. Die Einzelergebnisse der Antworten werden gemäß einer Zuordnungsvorschrift von Aussagen zu Rollen aufsummiert[1],

[1] Die genaue Zuordnung der Aussagen zu Rollen kann bei Bedarf dem Anhang, Kapitel B.3.2.2 entnommen werden.

so dass man zehn Ergebnisse erhält, die jeweils im Intervall [15 . . . 45] liegen. Fasst man diese Ergebnisse zusammen, so ergibt sich ein Rollenprofil, das Auskunft darüber gibt, in welchem Maß ein Teilnehmer zu den einzelnen Rollen tendiert oder auch nicht tendiert: Je höher die zu einer Rolle gehörige Punktzahl ist, desto größer ist die jeweilige Rollenausprägung des Teilnehmers und somit auch dessen Tendenz, diese Rolle auszuüben. Dies beinhaltet letztlich auch die Bereitschaft, die mit einer Rolle verbundenen Funktionen zu erfüllen und gleichsam zugehörige Rechte wie Pflichten wahrzunehmen.

9.2.2.2 Ausprägungen von VitaminL-Rollen

Der im Anhang, Kapitel B.3.2 vollständig wiedergegebene Fragebogen wurde innerhalb des VitaminL-Projekts elektronisch in Form eines Online-Fragebogens umgesetzt und auf einer Internet-Seite bereitgestellt. Da sich das VitaminL-Rollenmodell aus dem Rollenmodell von Spencer & Pruss herleitet, ist auch die Anwendung der Fragebogenergebnisse auf das VitaminL-Rollenmodell gerechtfertigt. Im Zuge dieser Anpassung wurden auch die Punktzahlen der gegebenen Antwortmöglichkeiten um je 1 reduziert, so dass die einzelnen Rollenausprägungen des VitaminL-Rollenmodells nunmehr jeweils im Intervall [0 . . . 30] liegen.

Sämtliche Teilnehmer der diversen Benutzertests wurden gebeten, den Fragebogen auszufüllen und damit ein Rollenprofil für Forschungszwecke im Rahmen des VitaminL-Projekts zur Verfügung zu stellen. Hierbei ist besonders zu betonen, dass die Probanden vor Ausfüllen des Fragebogens keinerlei Information über das zugehörige Rollenmodell oder über das Auswertungsprozedere des Fragebogens verfügten. Auf diese Weise wird verhindert, dass die Probanden beim Beantworten versuchen, eine wie auch immer geartete Erwartungshaltung zu erfüllen. Dieses Vorgehen wird auch von den Autoren des Fragebogens nahegelegt: *„Es wird nachdrücklich empfohlen, erst die Fragen zu beantworten und dann in der Tabelle am Ende des Fragebogens die Beschreibungen der Rollen nachzuschauen. Ebenso sollten, wenn dieser Fragebogen für Teambildungsübungen verwendet wird, die Rollenbeschreibungen ausgelassen werden, bis die Fragen beantwortet sind. Damit wird vermieden, dass die Fragen ungewollt mit einer bevorzugten Rolle im Hinterkopf beantwortet werden"* (Spencer & Pruss, 1995, S.75f).

Über den Projektzeitraum verteilt konnten auf diese Weise 177 Rollenprofile erfasst und – pseudonymisiert – in der in Kapitel 8.4.4.3.2 beschriebenen Wissensbasis hinterlegt werden (Stand: 01/2007). Sämtliche erfasste Rollenprofile können bei Bedarf der Tabelle C.1 im Anhang, Kapitel C.1 entnommen werden. Für jede Rolle ist das Intervall ihrer Ausprägungen in Abbildung 9.3 als horizontale Linie notiert, bei der das Minimum und Maximum mittels kleiner Endbalken markiert werden. Der (arithmetische) Mittelwert[2] einer Rollenausprägung ist durch einen Kreis auf der Line dargestellt.

[2] Auf eine zusätzliche Darstellung der geometrischen Mittelwerte wurde zugunsten einer besseren Lesbarkeit verzichtet.

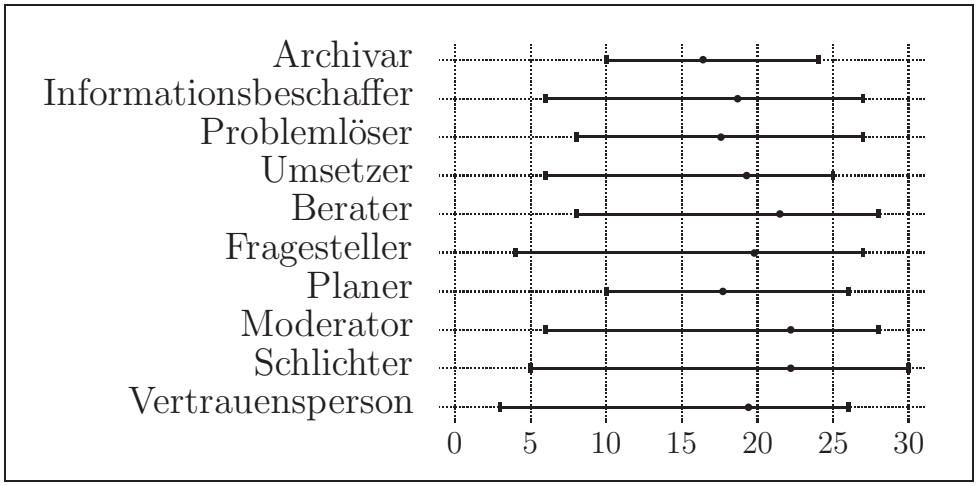

Abb. 9.3: Minima, Mittelwerte und Maxima der Rollenausprägungen

Insgesamt hinterlassen die Teilnehmer mit ihren Rollenprofilen ein sehr uneinheitliches Bild. Dieses äußert sich beispielsweise in sehr unterschiedlichen Profilen der Teilnehmer trotz einer an sich relativ homogenen Struktur der Teilnehmermenge (s. Kap.1.1 und Kap.7.1.1). Dabei unterscheiden sich nicht nur die einzelnen Teilnehmer voneinander in ihren Rollenprofilen (und den dort enthaltenen Rollenausprägungen; s. Tab.C.1), auch die einzelnen Rollen sind – über die Gesamtheit der Profile betrachtet – durchaus unterscheidlich vertreten. Neben unterschiedlichen Mittelwerten, die sich zwischen 16,4 und 22,4 Punkten (arithmetisches Mittel) beziehungsweise zwischen 16,1 und 22,0 Punkten (geometrisches Mittel) befinden, sind die Rollen auch mit stark voneinander abweichenden Bandbreiten ausgeprägt: Der Archivar ist mit Ausprägungen zwischen 10 (Minimum) und 24 (Maximum) Punkten besetzt, wohingegen beim Schlichter Ausprägungen von 5 bis 30 Punkten zu verzeichnen sind, um nur einige wenige Beispiele zu nennen. Details können bei Bedarf der Tabelle C.2 im Anhang, Kapitel C.1.2 entnommen werden.

9.3 Modellzusammenfassung

Insgesamt ist das Rollenmodell des VitaminL-Projekts als eine Interpretation des ursprünglichen Rollenmodells von Spencer & Pruss (1995) im Kontext der objektorientierten Programmierung von Java und dem bereits beschriebenen zugehörigen Lernprozess zu verstehen.

9.3.1 Rollen im Vorgehensmodell

Entsprechend der Charakterisierungen der einzelnen Rollen des VitaminL-Rollenmodells ergibt sich aus der Zuordnung der Rollen zu den Phasen (und Teilphasen) ein matrixähnliches Gesamtmodell, welchem die Beteiligung von Rollen

an einzelnen den Phasen des Vorgehensmodells entnommen werden kann. Das Ergebnis ist in den Tabellen 9.3 und 9.4 zusammenfassend dargestellt, beginnend mit der Zuordnung der VitaminL-Rollen zu den Elementen des technischen Prozesses.

Tab. 9.3: Rollen im fachlichen Prozess des VitaminL-Vorgehensmodells

	Archivar	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
Arbeitsvorbereitung										
A: Problemerkennung		×			×		×			
A.1 Analyse der Aufgabenstellung					+		+			
A.2 Klärung der Aufgabenstellung		+			+		+			
A.3 Identifikation von Teilproblemen		+			+		+			
B: Herstellung des Lernkontexts	○	×	×		×	○				
B.1 Zuordnung Probleme ← Lerninhalte		+			+					
B.2 Ermittlung von Lösungsansätzen	+	+	+		+					
B.3 Bewertung der Lösungsansätze			+		+	+				
B.4 Auswahl geeigneter Lösungsansätze			+		+					
C: Organisation der Problemlösung		×	×		×		×			
C.1 Definition von Teilaufgaben		+	+		+					
C.2 Verteilung der Aufgaben					+		+			
Umsetzung										
D: Codierung der Rohfassung	○	×	×	×	×					
D.1 Suche nach Templates	+	+			+					
D.2 Anpassung von Templates		+	+	+	+					
D.3 Integration von Templates in Lösung			+	+						
E: Fehlerbereinigung	○	○	×	×	○	○	○			
E.1 Übersetzung mit Compiler				+						
E.2 Erkennen von Compile-Fehlern				+	+					
E.3 Lokalisieren von Compile-Fehlern		+	+	+						
E.4 Maßnahme zur Fehlerbeseitigung finden	+	+	+		+	+				
E.5 Maßnahme zur Fehlerbeseitigung anwenden			+	+						
E.6 Ausführung mit Interpreter				+			+			
E.7 Erkennen von Laufzeitfehlern				+	+					
E.8 Lokalisierung von Laufzeitfehlern		+	+	+						
E.9 Maßnahme zur Beseitigung finden	+	+	+		+	+				
E.10 Maßnahme zur Beseitigung anwenden			+	+						

Fortsetzung auf nächster Seite

Tab. 9.3: Rollen im fachlichen Prozess des VitaminL-Vorgehensmodells *Forts.*

	Archivar	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
F: Abschlussarbeiten	×			×						
F.1 Formatierung des Quellcodes				+						
F.2 Kommentare	+			+						
Unterstützung										
X: Koordination	∘				×		×			
X.1 Einhaltung von Fristen und Terminen	+						+			
X.2 Koordination von Teilaufgaben					+		+			
X.3 Überprüfung von Ergebnissen					+		+			
Y: Informationsbeschaffung	×	×	×							
Y.1 Recherche	+	+	+							
Y.2 Nachfragen beim Tutor		+								
Z: Fachliche Kommunikation	∗	×	×	×	×	×	×			
Z.1 Fragen stellen				+		+				
Z.2 Erklärungen geben		+	+		+		+			
Z.3 Meinungen äußern		+			+	+	+			
Z.4 Ideen liefern		+	+	+	+		+			

Diese Tabelle ist wie folgt zu lesen: Das Symbol + in einem Feld markiert die Teilnahme einer Rolle (=Spalte) an einem Schritt des Vorgehensmodells (=Zeile), ein leeres Feld hingegen bedeutet, dass eine Rolle nicht (oder nicht nennenswert) an einem solchen Schritt teilnimmt. Diese Angaben werden für jede Phase des fachlichen Prozesses verdichtet: Falls eine Rolle an mindestens der Hälfte der Schritte einer Phase beteiligt ist, wird dies durch ein × in dem entsprechenden Feld einer Phase symbolisiert, ein ∘ markiert eine Teilnahme an weniger als 50 % der Schritte der Phase, ein leeres Feld bedeutet keine – oder keine nennenswerte – Teilnahme an der Phase. Das Symbol ∗ kennzeichnet eine prinzipielle Teilnahme des Archivars an sämtlicher fachlicher Kommunikation; dies ist jedoch für die Wahrnehmung seiner zweiten Funktion als Informationslieferant unentbehrlich und wird daher in der vorliegenden Tabelle nicht explizit ausgeführt, um somit eben diesen zweiten Aspekt des Archivars entsprechend zu betonen.

In ähnlicher Form wird die Teilnahme der Rollen an den Aktivitäten des sozialen Prozesses innerhalb der Tabelle 9.4 notiert. Aufgrund der bereits erwähnten vereinfachten Betrachtung der sozialen Aktivitäten ergibt sich auch hier eine kompaktere Form der Darstellung: Ein + markiert die Teilnahme einer Rolle an einer

sozialen Aktivität, ein leeres Feld bedeutet auch hier, dass die Rolle nicht an der zugehörigen Aktivität beteiligt ist. Eine Verdichtung wie in Tabelle 9.3 erfolgt hier jedoch nicht: Der soziale Prozess als solcher ist – im Gegensatz zum fachlichen Prozess – nicht in aufeinanderfolgende Elemente strukturiert, vielmehr werden die einzelnen Bestandteile des sozialen Prozesses als voneinander unabhängig begriffen, wobei Wechselwirkungen untereinander zwar auftreten können, aber für die in dieser Arbeit durchgeführten Betrachtungen nicht weiter von Bedeutung sind.

Tab. 9.4: Rollen im sozialen Prozess des VitaminL-Vorgehensmodells

	Archivar	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
Vermittlung in Konflikten						+		+	+	+
Motivation anderer Teilnehmer					+			+		
Intervention bei Offtopic-Aktivitäten					+		+	+	+	
Intervention bei Inaktivitäten					+	+		+		
Klärung von Kommunikationsproblemen						+		+	+	+

Insgesamt liefert die Synthese aus Vorgehensmodell und Rollenmodell ein Modell, welches Auskunft über die (idealerweise) an einem Element des Vorgehensmodell beteiligten Rollen – und damit auch erste Hinweise darauf, welche Rollen bei vorhandenen Defiziten hinsichtlich dieser Rollen in Problemsituationen gegebenenfalls durch das System zu kompensieren sind.

9.3.2 Kategorisierung der Rollen

Als ein weiteres Ergebnis des in diesem Kapitel definierten VitaminL-Arbeitsmodells, das sich aus der Kombination des Vorgehensmodells mit dem Rollenmodell von VitaminL ergibt, lässt sich eine Einteilung der Rollen in verschiedene Kategorien ableiten. Betrachtet man die in den Tabellen 9.3 und 9.4 zusammengefassten Ergebnisse, so lassen sich drei Rollenkategorien definieren.

9.3.2.1 Soziale Rollen

Mit dem Moderator, dem Schlichter und der Vertauensperson existieren im VitaminL-Rollenmodell drei Rollen, die ausschließlich im sozialen Prozess zum Einsatz kommen. Die Rollen beziehungsweise ihre Träger tragen bei auftretenden

Störungen zu deren Bereinigung bei, um somit eine reibungslose Fortführung des fachlichen Prozesses zu erreichen. Die sozialen Rollen des VitaminL-Rollenmodells finden sich in ähnlicher Form in den sozio-emotionalen Rollen nach Eunson wieder (s. Kap.2.5.4, Kap.B.2.2). Insgesamt wird diesen Rollen eine unterstützende Funktion hinsichtlich des fachlichen Prozesses – und der damit verbundenen Aufgabenbearbeitung – zugeschrieben, ohne jedoch direkt daran beteiligt zu sein.

9.3.2.2 Technische Rollen

Die technischen Rollen sind unmittelbar in den fachlichen Prozess involviert, das heißt die Inhaber dieser Rollen sind primär an der Aufgabenbearbeitung beteiligt. Diese Beteiligung umfasst den Umsetzer, den Problemlöser, den Informationsbeschaffer sowie den Archivar. Analog zu den Aufgabenrollen von Eunson (s. Kap.2.5.4, Kap.B.2.1) ist die Wahrnehmung dieser Rollen notwendig im Sinne der Aufgabenbearbeitung durch die Gruppe.

9.3.2.3 Integrierende Rollen

In der dritten Rollenkategorie finden sich diejenigen Rollen wieder, die sowohl am fachlichen als auch am sozialen Prozess beteiligt sind. Dazu zählen der Berater, der Fragesteller und der Planer. Die Beteiligung erfolgt in der Regel in Zusammenarbeit mit anderen Rollen aus den jeweiligen Prozessen. Die integrierenden Rollen stellen eine unterstützende Begleitung des fachlichen Prozesses dar, indem sie diesen durch organisatorische und beraterische Maßnahmen sowie durch soziale Aktivitäten aufrecht erhalten. Daran zeigt sich der integrierende Charakter dieser Rollen: Neben der Überwachung des fachlichen Prozesses tragen die Inhaber der integrierenden Rollen auch zu dessen Aufrechterhaltung durch geeignete Aktivitäten im sozialen Prozess bei.

Fasst man die technischen und die integrierenden Rollen zusammen, so ergeben sich ähnliche Konstellationen von Rollen, wie sie auch von Belbin (s. Kap.2.5.2), von Margerison und McCann (s. Kap.2.5.3) oder aber auch von Eunson mit dessen Aufgabenrollen (s. Kap.2.5.4, Kap.B.2.1) identifiziert wurden.

9.3.3 Gewichtung der Rollen

Wie aus den Ausführungen des Kapitels 9.2.1 ersichtlich ist, werden einzelnen Rollen unterschiedliche Bedeutungen für das Vorgehensmodell und den damit assoziierten Lernprozess beigemessen. Dementsprechend sind einige Rollen wichtiger als andere Rollen anzusehen, Rollen wie die Vertrauensperson (s. Kap.9.2.1.10) sind sogar gänzlich entbehrlich. Im Hinblick auf eine angemessene tutorielle Unterstützung eines Teams sollte somit auch die Relevanz der einzelnen Rollen für den Lernprozess Berücksichtigung finden. Dazu ist zu klären, wie die einzelnen Rollen des Rollenmodells im Hinblick auf ihre Relevanz für den Lernprozess zu gewichten sind.

9.3.3.1 Relevanz der Rollen für den Lernprozess

Die Bedeutung einer Rolle für den Lernprozess drückt sich zu einem gewissen Teil auch an ihrer Beteiligung von Rollen an Schritten und Phasen des Vorgehensmodells aus, die in den vorangegangenen Tabellen 9.3 und 9.4 wiedergegeben wird. Die nun folgende Tabelle 9.5 fasst diese Zuordnungen von Rollen zu Schritten und Phasen des Vorgehensmodells zusammen und liefert eine rein quantitative Beschreibung dieses Gesamtmodells.

Tab. 9.5: Quantitative Betrachtung von Rollen im Vorgehensmodell

	Archivar	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
Beteiligung an Schritten (+)	5	16	15	14	23	8	11	5	3	2
Hohe Beteiligung an Phasen (×)	0	6	5	4	5	1	4	0	0	0
Geringe Beteiligung an Phasen (○)	5	1	1	0	2	2	1	0	0	0

Der Informationsbeschaffung wird im Lernprozess ein sehr hoher Stellenwert eingeräumt (s. Kap.7.3.2.1). Dies spiegelt sich auch in der Tabelle 9.5 wieder: Der Informationsbeschaffer ist an 16 Schritten des Vorgehensmodells beteiligt, was einem zweiten Platz (nach dem Berater mit 23 Beteiligungspunkten) entspricht. Hinzu kommen beim Informationsbeschaffer sechs Phasen mit hoher Beteiligung und eine Phase mit geringer Beteiligung. Insgesamt wird dem Informationsbeschaffer – trotz höherer Vergleichswerte des Beraters bei rein quantitativer Betrachtung – eine deutlich höhere Relevanz für den hier behandelten Lernprozess eingeräumt, zumal der Berater eine eher unterstützende Funktion ausübt.

Der Planer würde bei zahlenmäßiger Betrachtung der erreichten Beteiligungspunkte in der Relevanz nur Rang 4 bis 5 belegen, jedoch ist zu bedenken, dass auch die Berücksichtigung übergeordneter Ziele bei der Aufgabenbearbeitung unerlässlich ist. Daher wird der Planer hinsichtlich seiner Relevanz für den Lernprozess letztlich höher bewertet und landet – direkt hinter dem Informationsbeschaffer – auf Rang 2, gefolgt vom Berater.

Auf Rang 4 wird der Umsetzer eingestuft, welcher die praktischen Fertigkeiten als unentbehrlichen Bestandteil der Programmierung und des damit verbundenen Lernprozesses repräsentiert. Dies entspricht auch einer Einstufung nach erreichten Beteiligungspunkten: Sowohl bei den Schritten (Beteiligun an 14 Schritten) als auch bei den Phasen (vier Phasen mit hoher Beteiligung) wird dem Umsetzer jeweils der vierte Platz zugewiesen.

Betrachtet man die weiteren Werte in der Tabelle 9.5, so müsste der Schlichter im Prinzip auf einem der hinteren Plätze positioniert werden: Die Beteiligung an drei Schritten des Vorgehensmodells entspricht einem 9. Platz. Im Gegensatz dazu haben die Benutzertests jedoch gezeigt, dass durchaus Bedarf an einer konfliktbereinigenden Rolle wie dem Schlichter besteht (s. Kap.9.1.2.4). Entsprechend dieses Bedarfs ist die Rolle des Schlichters wesentlich höher einzustufen und wird infolgedessen mit einem 5. Platz versehen.

Wie bereits in Kapitel 9.2.1.10 ausgeführt, besteht für die Rolle der Vertrauensperson aufgrund der Ausgestaltung der virtuellen Teamarbeit durch das VitaminL-System und der damit einhergehenden fehlenden Unterstützung kein Bedarf. Hinsichtlich ihrer Relevanz für den Lernprozess landet die Vertrauensperson folglich auf dem letzten Platz. Dies entspricht auch der Einordnung auf Basis der in Tabelle 9.5 erreichten Beteiligungspunkte.

Die Funktionalität des Archivars wird – wie bereits in Kapitel 9.2.1.9 erläutert – weitgehend durch das VitaminL-System erfüllt, so dass eine weitere Unterstützung dieser Rolle durch die tutorielle Komponente des VitaminL-Systems nicht erforderlich ist. Der Archivar wird damit auf den 9. Platz verwiesen.

Ebenfalls auf einem der hinteren Plätze hinsichtlich seiner Relevanz für den Lernprozess ist der Moderator einzuordnen: Diese Rolle wird als externes Teammitglied nur im Bedarfsfall aktiviert. Damit wird der Moderator auf Platz 8 eingestuft. Die in den verschiedenen Kategorien erreichten Beteiligungspunkte untermauern diese Bewertung.

Auf die beiden übriggebliebenen Plätze 6 und 7 im Mittelfeld der Relevanzskala werden entsprechend den in Tabelle 9.5 erreichten Beteiligungspunkten der Problemlöser (Plaz 6) und der Fragesteller (Platz 7) eingestuft.

Tab. 9.6: Relevanzrangfolge der VitaminL-Rollen

Rang	**VitaminL-Rolle**	**Kürzel**
1	Informationsbeschaffer	IB
2	Planer	PL
3	Berater	BR
4	Umsetzer	UM
5	Schlichter	SL
6	Problemlöser	PR
7	Fragesteller	FR
8	Moderator	MD
9	Archivar	AR
10	Vertrauensperson	VP

Tabelle 9.6 fasst diese Überlegungen zusammen und stellt die einzelnen Rollen des VitaminL-Rollenmodells in der Reihenfolge ihrer jeweiligen Bedeutung für den

Lernprozess dar. Da nun die Rollen auf den oberen Rängen eine höhere Relevanz für den betrachteten Lernprozess besitzen als diejenigen auf darunterliegenden Rängen, ist bei der Ausgestaltung der Tutorkomponente darauf zu achten, dass Defizite von Rollen – insbesondere von solchen mit hoher Relevanz – innerhalb eines virtuellen Teams bei dessen Zusammenarbeit nicht nur erkannt, sondern durch das System auch angemessen kompensiert werden.

9.3.3.2 Relevenz der Rollen aus Sicht der Teilnehmer

Begleitend zu einer Phase von Benutzertests, die im Dezember 2005 stattfanden, wurden die Teilnehmer zu ihrer persönlichen Einstätzung der Rollen befragt: Zu jeder der zehn Rollen enthielt der ausgehändigte Fragebogen eine kurze Charakterisierung der jeweiligen Rolle sowie drei Fragen:

1. *„In welchem Umfang hat Ihnen diese Teamfunktion für die erfolgreiche Bearbeitung Ihrer Aufgabe gefehlt?"*
2. *„Wie wichtig ist Ihrer Meinung nach diese Teamfunktion für den Gruppenerfolg?"*
3. *„In welchem Umfang erfüllen Sie selbst Ihrer Meinung nach diese Teamfunktion?"*

Jede dieser Fragen konnte mittels einer sechselementigen Skala beantwortet werden, oberhalb derer von links nach rechts zur Orientierung die Antwortmöglichkeiten *„sehr"* (links), *„wenig"* (mittig) und *„gar nicht"* (rechts) notiert waren. Die Antworten wurden zur quantitativen Auswertung auf die Schulnoten von `1` (*„sehr"*) bis `6` (*„gar nicht"*) abgebildet. Als weitere Antwortmöglichkeit wurde *„weiss nicht"* angeboten: Diese Antworten wurden bei der nachfolgenden Auswertung nicht berücksichtigt.

Insgesamt 27 von 34 Teilnehmern beantworteten den Fragebogen[3]. Einer der Teilnehmer beantwortete jedoch die Fragen zur Einschätzung der Rollen nicht, so dass nur 26 Einschätzungen hinsichtlich der Relevanz der einzelnen Rollen zur Auswertung herangezogen werden können. Die nachfolgende Tabelle 9.7 fasst die Ergebnisse der Antworten der ausgefüllten Fragebögen zur Frage nach der Wichtigkeit der jeweiligen Rolle zusammen und stellt diese den Ergebnissen des vorigen Kapitels 9.3.3.1 in einem Vergleich gegenüber:

1. Die erste Spalte (*VitaminL-Rolle*) enthält die zehn VitaminL-Rollen in der Reihenfolge ihrer Relevanz gemäß Tabelle 9.6 in Kapitel 9.3.3.1 (Spalte *VitaminL-Rang*).
2. In der zweiten Spalte (ϕ*-Note*) sind die Antworten in Form eines Notendurchschnitts (arithmetischer Mittelwert) wiedergegeben: Je kleiner ein Wert

[3] Dies entspricht einer Rücklaufquote von 79,41 %.

ist, desto bedeutsamer wird die zugehörige Rolle von den Teilnehmern eingeschätzt.

3. Die sich aus dieser Bewertung ergebende Reihenfolge wird in der dritten Spalte (*Teilnehmer-Rang*) dargestellt.

4. Die vierte Spalte (*Vergleich*) vergleicht die beiden Rangfolgen der Spalten 3 und 5 miteinander.

Die einzelnen Ergebnisse dieser Umfrage sind in der Tabelle B.6 im Anhang, Kapitel B.4.2 aufgeführt.

Tab. 9.7: Bewertung der VitaminL-Rollen durch Teilnehmer

VitaminL-Rolle	**ϕ-Note**	**Teilnehmer-Rang**	**Vergleich**	**VitaminL-Rang**
Informationsbeschaffer	2,3	3	$>$	1
Planer	2,0	2	$=$	2
Berater	2,6	5	$>$	3
Umsetzer	2,5	4	$=$	4
Schlichter	3,0	7	$>$	5
Problemlöser	1,8	1	$\ll$	6
Fragesteller	2,7	6	$<$	7
Moderator	3,2	8	$=$	8
Archivar	3,5	9	$=$	9
Vertrauensperson	4,3	10	$=$	10

Die durch Vergleich der verschiedenen Rangfolgen in den Spalten 3 (*Teilnehmer-Rang*) und 5 (*VitaminL-Rang*) erzielten Ergebnisse sind in Spalte 4 dargestellt:

- Identische Werte in beiden Rangfolgen werden mit = notiert.

- Unterscheiden sich die beiden Ränge einer Rolle um höchstens zwei Positionen voneinander, so wird dies mit $<$ bzw. $>$ notiert und als geringe Abweichung interpretiert.

- Felder mit $\ll$ oder $\gg$ kennzeichnen größere Unterschiede in den beiden Rängen einer Rolle und werden auch als nicht-geringe Abweichung interpretiert.

Vergleicht man die Ergebnisse der Spalten 3 und 5 miteinander wie in Spalte 4 dargestellt, so kann weitestgehend Übereinstimmung festgestellt werden: Die Hälfte der Rollen (Planer, Umsetzer, Moderator, Archivar und Vertrauensperson) wird von den Teilnehmern mit demselben Rang wie auch im VitaminL-Modell bewertet, bei weiteren vier Rollen (Informationsbeschaffung, Berater, Schlichter und Fragesteller) wird eine nur geringe Abweichung um höchstens zwei Positionen festgestellt, lediglich eine Rolle (Problemlöser) wird gänzlich unterschiedlich eingestuft (Rang 6 im VitaminL-Modell vs. Rang 1 nach Einschätzung). Der Grund für diese große Abweichung ist offensichtlich, denn was ist aus der Sicht von jemandem, der eine Aufgabe zu erledigen hat, reizvoller als jemand, der unliebsame oder verzwickte Tätigkeiten übernimmt? So kann aus dem Blickwinkel der Teilnehmer auch der Problemlöser betrachtet werden.

Insgesamt kann diese Umfrage – trotz der hohen Abweichung bei der Rolle des Problemlösers – durchaus als Bestätigung des in Kapitel 9.3.3.1 erarbeiteten Ergebnisses angesehen werden.

Kapitel 10

Rollenanalyse

In diesem Kapitel wird die in Kapitel 7.3.4.2 formulierte Fragestellung zur Zusammensetzung virtueller Teams und deren Erkennung behandelt. Das Ziel besteht darin, ein auf statistischen Methoden basierendes Verfahren zu entwickeln, das in der Lage ist, bereits während einer laufenden Sitzung für alle beteiligten Teammitgliedern deren Grad an der Zugehörigkeit zu den einzelnen Rollen des VitaminL-Rollenmodells (s. Kap.9.2) zu bestimmen. Dazu werden sämtliche Aktionen aller Teammitglieder zentral auf dem Vitaminl-Server erfasst und an die Tutorkomponente zur weiteren Verarbeitung weitergeleitet. Innerhalb der Tutorkomponente werden die CLS++-Codes der Benutzeraktionen (s. Kap.8.4.3) zur Rollenanalyse herangezogen.

Prinzipiell lassen sich die Rollenprofile von Teilnehmern auch a priori durch Ausfüllen des Fragebogens nach Spencer & Pruss ermitteln. Dieses Vorgehen ist jedoch einerseits mit einem gewissen Zusatzaufwand verbunden[1] und erfolgt andererseits auch losgelöst vom jeweiligen Anwendungskontext. Daher wird die Entwicklung eines Verfahrens zur Rollenbestimmung während der (synchronen) Zusammenarbeit bei der Programmierung in Java angestrebt.

10.1 Vorgehensweise

Auf der Grundlage bereits abgeschlossener und somit vollständig protokollierter Sitzungen kommen – in Verbindung mit bereits erfassten Rollenprofilen der betreffenden Teilnehmer – statistische Analyseverfahren zum Einsatz, mit denen das vorhandene Datenmaterial auf – zunächst lineare – Beziehungen hin untersucht werden soll. Mit der Auswahl eines quantitativen Analyseverfahrens sieht sich die vorliegende Arbeit in einer Linie mit Verfahren wie IPA (s. Kap.2.4.4.1.1) und SYMLOG (s. Kap.2.4.4.1.2). Der ursprüngliche Nachteil dieser Verfahren, der mit einem verhältnismäßig hohen Aufwand bei der Durchführung benannt wird (s. Kap.2.4.4.1, Kap.2.4.4.2) wird durch den Einsatz moderner Informations- und

[1] Der Teamfragebogen von Spencer & Pruss umfasst 150 zu bewertende Aussagen (s. Kap.2.5.5, 9.2.2.1 und B.3.2).

Kommunikationstechnologien und die daraus resultierende Möglichkeit der maschinellen Verarbeitung kompensiert.

Im Anschluss an die Analyse erfolgt die Umsetzung verwertbarer Ergebnisse: Diejenigen Beziehungen, die sich durch die Analyse statistisch erklären lassen, werden innerhalb der Tutorkomponente des VitaminL-Systems derart realisiert, dass während einer laufenden Sitzung die Benutzeraktionen – ähnlich wie bei deren Protokollierung – erfasst und entsprechend den Analyseergebnissen in Kennzahlen umgesetzt werden. Diese Kennzahlen dienen als Indikatoren für die Rollenausprägungen der teilnehmenden Benutzer und werden innerhalb der Tutorkomponente weitergereicht mit dem Ziel, geeignete Hilfeunterstützung in Problemsituationen zu generieren (vgl. KÖLLE, 2007). Zu diesem Zwecke ist die in Kapitel 8.4.5 skizzierte Architektur geeignet zu erweitern. Eine abschließende Evaluation auf der Basis weiterer Benutzertests dient der Überprüfung und Bewertung der umgesetzten Ergebnisse der statistischen Analyse.

10.1.1 Datenbasis der Rollenanalyse

Die Durchführung der geplanten Analyse beruht auf Daten von Teilnehmern, die einerseits den in Kapitel 9.2.2.1 vorgestellten Fragebogen zur Ermittlung des Rollenprofils ausgefüllt haben und andererseits an mindestens einem Benutzertest teilgenommen haben. Zu diesem Zwecke wurde zwischen Dezember 2004 und März 2005 eine umfangreiche Datenerhebung durchgeführt.

10.1.1.1 Verhaltensdaten

Im Rahmen dieser Datenerhebung fanden Benutzertests zur Sammlung von Sitzungsdaten statt.

10.1.1.1.1 Verteilung der Teilnehmer auf Sitzungen An diesen Benutzertests beteiligten sich insgesamt 31 Studierende, die – verteilt auf elf Gruppen – an 25 Sitzungen teilnahmen. Die nachfolgende Tabelle 10.1 gibt die genaue Verteilung der einzelnen Teilnehmer auf die einzelnen Gruppen und Sitzungen wieder. Die jeweilige Sitzungsdauer ist in Minuten angegeben. Insgesamt ergeben sich 73 Teilnahmen von Teilnehmern an Sitzungen.

Tab. 10.1: Benutzertests zur Vorbereitung der Rollenanalyse

Sitzung	Teilnehmer	Gruppe	Anmerkung	Dauer
S_1	T_4, T_5, T_6	G_1	Test 1	67
S_2	T_7, T_8, T_9	G_2	Test 1	97
S_3	T_{11}, T_{12} / T_{10}	G_3	Test 1	94
S_4	T_{13}, T_{14}, T_{15}	G_4	Test 1	91

Fortsetzung auf nächster Seite

Tab. 10.1: Benutzertests zur Vorbereitung der Rollenanalyse *Forts.*

Sitzung	**Teilnehmer**	**Gruppe**	**Anmerkung**	**Dauer**
S_5	T_{19}, T_{20}, T_{21}	G_5	Test 1	98
S_6	T_{22}, T_{23}, T_{25}	G_6	Test 1	81
S_7	T_{29}, T_{30}, T_{31}	G_7	Test 1	99
S_9	T_{26}, T_{27}, T_{28}	G_8	Test 1	90
S_{10}	T_{16}, T_{17}, T_{18}	G_9	Test 1	91
S_{11}	T_{32}, T_{33}, T_{34}	G_{10}	Test 1	73
S_{12}	T_4, T_5, T_6	G_1	Test 2	116
S_{13}	T_{29}, T_{30}, T_{31}	G_7	Test 2	100
S_{14}	T_{26}, T_{27}, T_{28}	G_8	Test 2	116
S_{15}	T_{22}, T_{25}	G_6	Test 2	109
S_{17}	T_6, T_{11}, T_{12} / T_{10}	G_3	Test 2	101
S_{18}	T_7, T_8, T_9	G_2	Test 2	108
S_{19}	T_{19}, T_{20}, T_{21}	G_5	Test 2	109
S_{20}	T_{32}, T_{33}, T_{34}	G_{11}	Test 2	88
S_{20}	T_{32}, T_{33}, T_{34}	G_{11}	Test 2	88
S_{21}	T_{29}, T_{30}, T_{31}	G_7	Test 3	105
S_{22}	T_4, T_5, T_6	G_1	Test 3	105
S_{23}	T_{19}, T_{20}, T_{21}	G_5	Test 3	102
S_{24}	T_{22}, T_{25}	G_6	Test 3	102
S_{25}	T_{32}, T_{33}, T_{34}	G_{11}	Test 3	89
S_{26}	T_{32}, T_{33}, T_{34}	G_{11}	Test 4	151
S_{27}	T_{22}, T_{25} / T_{24}	G_6	Test 4	126
25	29 / 2	11		$\phi_{ari} = 100,32$

Die Häufigkeit der Teilnahme ist in Abbildung 10.1 zusammengefasst.

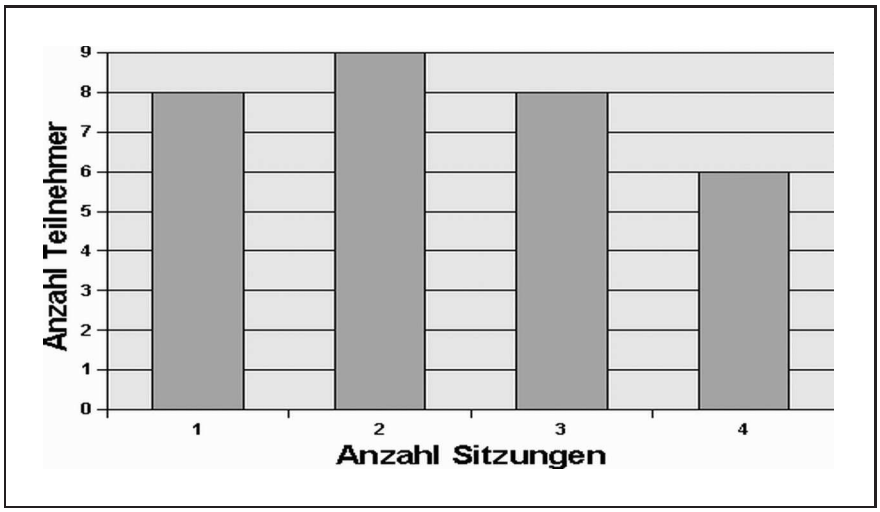

Abb. 10.1: Verteilung der Teilnehmer von Benutzertests (S_1-S_{27})

Sehr gut zu erkennen ist die Tatsache, dass erstmalig Teilnehmer mehrere Sitzungen absolvierten: Acht Studierende nahmen an nur einer Sitzung teil, neun Teilnehmer beteiligten sich an zwei Sitzungen, weitere acht Teilnehmer absolvierten drei Sitzungen und immerhin sechs der Probanden waren an vier Sitzungen beteiligt. Insgesamt haben demnach 23 der 31 Teilnehmer (=74,2 %) mehr als einmal mit dem VitaminL-System gearbeitet. Entsprechend reduzierte sich deren Einarbeitungsaufwand ab der zweiten Sitzung.

10.1.1.1.2 Aufgabenbeschreibungen Die Aufgabenstellungen, die in den in der vorigen Tabelle 10.1 mit *Test 1*, *Test 2* und *Test 3* kommentierten Sitzungen bearbeitet werden sollten, waren an die jeweils zuvor behandelten Lehrinhalte angepasst und wiesen infolgedessen einen leicht ansteigenden Schwierigkeitsgrad auf. Sie waren darüber hinaus so konzipiert, dass sie bereits klar definierte Teilaufgaben enthielten mit dem Ziel, den Teilnehmern die Verteilung von Aufgaben so einfach wie möglich zu gestalten, um den Schwerpunkt der Sitzungen auf die eigentliche Umsetzungsphase setzen zu können. Im vierten Durchlauf erhielten die Teilnehmer die Möglichkeit, ein konkretes Problem ihrer jeweiligen Abschlussarbeit im VitaminL-System zu bearbeiten. Dementsprechend waren die Aufgabenstellungen des vierten Durchlaufs individuell an die Projekte der jeweiligen Gruppen angepasst, wohingegen die Aufgaben der Durchläufe 1 bis 3 für alle Gruppen identisch gestaltet wurden.

10.1.1.1.3 Ablauf der Sitzungen Die einzelnen Sitzungen wurden weitestgehend wie in Kapitel 8.1.2.2 beschrieben durchgeführt, das heißt auf eine kurze Einleitung und Verteilung der Aufgabenstellung erfolgte die eigentliche Aufgabenbearbeitung. Nach einer kurzen Abschlussbesprechung, die sowohl die Teilnehmer als auch die Versuchsleitung für ein erstes Feedback nutzen konnten, endete jede Sitzung. Da viele der Teilnehmer sich an mehr als einem Benutzertest beteiligten, konnte bei diesen ab der zweiten Sitzung die Einleitung entfallen, so dass sich bei diesen die Gesamtdauer einer Sitzung entsprechend reduzierte. Wie der Spalte *Dauer* der Tabelle 10.1 entnommen werden kann, betrug die durchschnittliche Sitzungsdauer bei den ersten Sitzungen aller Gruppen (S_1-S_{11}) 88,1 Minuten. Die nachfolgenden Sitzungen dauerten in der Regel länger, insgesamt ergibt sich eine durchschnittliche Sitzungsdauer von 100,32 Minuten bei Betrachtung aller 25 Sitzungen. Insbesondere die letzten beiden Sitzungen (S_{26} und S_{27}) nehmen mit Dauern von 151 und 126 Minuten eine besondere Position ein: Einerseits haben die entsprechenden Gruppen damit zum jeweils vierten Mal mit dem VitaminL-System gearbeitet und damit eine gewisse Routine im Umgang mit der Software, andererseits erhielten die Gruppen in diesem vierten Durchgang die Gelegenheit, an ihrem jeweiligen Abschlussprojekt weiterzuarbeiten, so dass davon ausgegangen werden kann, dass die Motivation entsprechend höher war als bei der Bearbeitung von Hausaufgaben in den ersten drei Durchgängen.

Alle Sitzungen wurden in XML-Dateien protokolliert und in die in Kapitel 8.4.4.3.2 vorgestellte Wissensbasis eingepflegt.

10.1.1.2 Rollenprofile

Ergänzt wurde die Datenerhebung durch den Rollenfragebogen, den 29 von 31 Teilnehmern (= 93,5 %) ausgefüllt haben. Die 29 resultierenden Rollenprofile sind in der Tabelle 10.1 im Anhang, Kapitel C.2 zusammengefasst. Stellt man diese Stichprobe (N=29) der Gesamtheit der erfassten Rollenprofile (N=177; s. Kap.C.1) gegenüber, indem man beispielsweise die (arithmetischen) Mittelwerte der einzelnen Rollenausprägungen miteinander vergleicht, so stellt man fest, dass die Stichprobe in nur geringem Umfang von der Gesamtheit abweicht.

Tab. 10.2: Statische Betrachtungen der erfassten Rollenprofile

	Archivar	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
$\phi_{ari,29}$	17,14	19,69	18,48	19,66	22,10	19,93	18,48	22,31	22,28	19,17
$\phi_{ari,177}$	16,36	18,70	17,60	19,33	21,50	19,79	17,70	22,19	22,21	19,41
Δ_{abs}	0,78	0,99	0,88	0,32	0,61	0,14	0,78	0,12	0,06	-0,24
Δ_{rel}	4,78	5,29	4,99	1,66	2,82	0,71	4,42	0,56	0,28	-1,24

In der vorliegenden Tabelle 10.2 sind neben den arithmetischen Mittelwerten der einzelnen Rollenausprägungen sowohl für die genannte Stichprobe (Zeile $\phi_{ari,29}$) als auch für die Gesamtheit aller Rollenprofile (Zeile $\phi_{ari,177}$) enthalten. In der dritten Zeile (Δ_{abs}) ist die jeweilige (absolute) Differenz ($\phi_{ari,29}$-$\phi_{ari,177}$) dargestellt, in der letzten Zeile wird die Differenz als Relativwert berechnet, wobei die Gesamtheit ($\phi_{ari,177}$) den Bezugswert von 100 % bildet. Alle Werte sind hier auf zwei Nachkommastellen genau dargestellt, trotzdem sind kleine Ungenauigkeiten, bedingt durch Rundungsfehler, zu verzeichnen; diese können jedoch toleriert werden.

10.1.1.3 Aufbereitung der Datenbasis

Aus den aufgezeichneten Protokollen ergibt sich – in Kombination mit den Rollenprofilen der Teilnehmer – eine Vielzahl an Eingangsdaten, die vor der Analyse geeignet aufbereitet werden, um zur Klärung der Fragestellung herangezogen werden zu können. Diese Aufbereitung beinhaltet eine Abbildung sämtlicher Teilnehmeraktionen auf die in Kapitel 8.4.3.2 definierten CLS++-Codes. Dies bedeutet eine Reduktion sämtlicher Interaktionen der Teilnehmer mit dem System auf Tupel der Form (T_k, c_l), wobei T_k ein beliebiger Teilnehmer und c_l ein Code-Wert

aus CLS++ ist und eine von diesem Teilnehmer ausgelöste Aktion vom entsprechenden Typ repräsentiert (s. Kapitel 8.4.3). Im Zuge dieser Reduktion wird auf zusätzliche Informationen wie Inhalte von Kommunikationsbeiträgen, Zeilennummern bei Navigationsoperationen oder betroffene Dokumente bei Dateioperationen bewusst verzichtet.

10.1.1.3.1 Formale Begriffsdefinitionen Zur exakten Problemformulierung werden daher einige Begriffe formal festgelegt. Zunächst sei die Menge $\mathcal{T}$ aller Teilnehmer definiert als

$$\mathcal{T} = \{T_k \mid k > 0 \wedge T_k \text{ ist Teilnehmer}\} \tag{10.1}$$

und die Menge $\mathcal{C}$ aller (CLS++-)Codes sei

$$\mathcal{C} = \{C_l \mid 0 \leq l \leq 52 \wedge C_l \text{ ist Code einer Aktion gemäß CLS++}\}. \tag{10.2}$$

Eine Sitzung S_i wird definiert als

$$S_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,n_i}) \tag{10.3}$$

mit $s_{i,j} = (T_k, c_l) \in \mathcal{T} \times \mathcal{C}$, wobei $T_k \in \mathcal{T}$ einer der an der Sitzung S_i beteiligten Teilnehmer ist und $c_l \in \mathcal{C}$ der Code einer vom Teilnehmer T_k während dieser Sitzung ausgelösten Aktion. Zur Vorbereitung der späteren Analyse werden die Daten einer Sitzung aufbereitet, indem die Aktionencodes den auslösenden Teilnehmern zugeordnet und durch Kumulation verdichtet werden

Die Dauer einer Sitzung S_i in Minuten wird mit t_i bezeichnet. Pro Sitzung S_i und Teilnehmer T_k ergibt sich ein Code-Vektor $C_{i,k}$ mit

$$C_{i,k} = (C_{i,k,0}, C_{i,k,1}, \ldots, C_{i,k,52}), \tag{10.4}$$

wobei jedes enthaltene Element $C_{i,k,l}$ als *kumulierte Code-Kennzahl* die absolute Häufigkeit der von Teilnehmer T_k in der Sitzung S_i ausgelösten Aktion c_l darstellt:

$$C_{i,k,l} = \|\{s_{i,j} | s_{i,j} = (T_k, c_l) \in S_i\}\|. \tag{10.5}$$

Die Beteiligung oder Akktivität $A_{i,k}$ eines Teilnehmers an der Sitzung lässt sich als Summe der von ihm ausgelösten Aktionen wie folgt definieren:

$$A_{i,k} = \sum_{l=0}^{52} C_{i,k,l}. \tag{10.6}$$

Um die so definierten absoluten Sitzungsdaten von etwaigen Störeffekten zu bereinigen, die sich durch voneinander abweichende Sitzungsdauern oder unterschiedliche Beteiligungen einzelner Teilnehmer ergeben, wird jeder Code-Vektor $C_{i,k}$ transformiert in einen Code-pro-Zeit-Vektor $C^T_{i,k}$ mit

$$C^T_{i,k} = \left(\frac{C_{i,k,0}}{t_i \frac{1}{100}} , \frac{C_{i,k,1}}{t_i \frac{1}{100}} , \ldots , \frac{C_{i,k,52}}{t_i \frac{1}{100}} \right) \tag{10.7}$$

$$= \left(C^T_{i,k,0} , C^T_{i,k,1} , \ldots , C^T_{i,k,52} \right) \tag{10.8}$$

sowie in einen Code-Beteiligung-Vektor $C^B_{i,k}$ mit

$$C^B_{i,k} = \left(\frac{C_{i,k,0}}{A_{i,k}\frac{1}{100}} , \frac{C_{i,k,1}}{A_{i,k}\frac{1}{100}} , \ldots , \frac{C_{i,k,52}}{A_{i,k}\frac{1}{100}} \right) \quad (10.9)$$

$$= \left(C^B_{i,k,0} , C^B_{i,k,1} , \ldots , C^B_{i,k,52} \right). \quad (10.10)$$

Im Gegensatz zu den mittels Kumulation gebildeten Absolutwerten eines Code-Vektors $C_{i,k}$ enthalten die soeben definierten Varianten relative Werte, die sich entweder auf die Sitzungsdauer oder auf die Beteiligung an einer Sitzung beziehen. Pro Sitzung und Teilnehmer ergeben sich somit drei zu untersuchende Code-Vektoren, die nachfolgend zwecks eindeutiger Bennenung auch als *kumulierter Code-Vektor* ($C_{i,k}$), *zeitbezogener Code-Vektor* ($C^T_{i,k}$) und *beteiligungsbezogener Code-Vektor* ()$C^B_{i,k}$) bezeichnet werden. Die enthaltenen Elemente werden dementsprechend auch als *kumulierte Code-Kennzahlen* ($C_{i,k,l}$), als *zeitbezogene Code-Kennzahlen* ($C^T_{i,k,l}$) und als *beteiligungsbezogene Code-Kennzahlen* ($C^B_{i,k,l}$) benannt. Der in die jeweiligen Berechnungen eingehende Faktor $\frac{1}{100}$ (s. Formeln 10.7 und 10.9) dient lediglich der optischen Korrektur der berechneten Werte[2], auf die eigentlichen Ergebnissen und deren Aussagekraft hat dieser Korrekturfaktor keinerlei Einfluss.

Es sei außerdem die Menge $\mathcal{R}$ aller Rollen festgelegt als

$$\mathcal{R} = \{R_n \mid 1 \leq n \leq 10 \wedge R_n \text{ ist n-te VitaminL-Rolle}\}, \quad (10.11)$$

wobei für die Numerierung der Rollen die in Tabelle 9.6 definierte Rangfolge gilt, das heißt Rolle R_1 = Informationsbeschaffer, Rolle R_2 = Planer, ..., Rolle R_{10} = Vertrauensperson. Ferner wird das Rollenprofil P_k eines Teilnehmers T_k definiert als

$$P_k = (p_{k,1}, p_{k,2}, \ldots, p_{k,10}), \quad (10.12)$$

wobei $p_{k,i}$ die Ausprägung des Teilnehmers hinsichtlich der Rolle R_n darstellt. Die Menge $\mathcal{P}$ der Rollenprofile aller Teilnehmer ist damit definiert durch

$$\mathcal{P} = \bigcup_{k>0} \{P_k\}. \quad (10.13)$$

Somit lässt sich die Menge aller Ausprägungen einer Rolle R_n (mit $1 \leq n \leq 10$) festlegen als

$$\mathcal{P}_n = \{p_{k,n} | p_{k,n} \in P_k \wedge P_k \in \mathcal{P}\}. \quad (10.14)$$

[2] Durch Multiplikation mit dem Faktor 100 verringert sich die Anzahl darzustellender Nachkommastellen, so dass bei der Darstellung der Ergebnisse eine insgesamt bessere Lesbarkeit verzeichnet werden kann.

10.1.1.3.2 Beschreibung der Analysedaten In den 25 für die Analyse erfassten Sitzungen haben die daran beteiligten 26 Benutzer[3] insgesamt 77.727 Aktionen ausgelöst. Diese lassen sich für jede der drei Varianten ($C_{i,k}$, $C^T_{i,k}$ und $C^B_{i,k}$) in 70 Vektoren mit jeweils 53 verdichten Code-Kennzahlen organisieren. Je nach zugrundeliegenden Code-Vektoren werden die zusammengehörigen Daten über alle untersuchten Sitzungen als *kumulierte Sitzungsdaten*, *zeitbezogene Sitzungsdaten* und *beteiligungsbezogene Sitzungsdaten* bezeichnet. Über die drei Kategorien verfügbarer Sitzungsdaten ergeben sich für alle 53 CLS++-Codes insgesamt 11.130 (=70·3·53) quantitative Aussagen hinsichtlich deren Verwendung. Diese sind in Kombination mit den vorliegenden Rollenprofilen zu untersuchen.

Die diversen Sitzungsdaten lassen sich tabellarisch darstellen, so dass sich für jede der zehn zu betrachtenden Rollen zunächst drei Tabellen generieren lassen, von denen jeweils eine die kumulierten Sitzungsdaten, die zeitbezogenen beziehungsweise die beteiligungsbezogenen Sitzungsdaten enthält.

Die Tabelle 10.3 beschreibt formal die 3.710 (=70·53) kumulierten Sitzungsdaten, die Tabellen der zeitbezogenen und der beteiligungsbezogenen Sitzungsdaten sind analog aufgebaut.

Tab. 10.3: Formaler Aufbau der kumulierten Sitzungsdaten

Ausprägung	**Teilnehmer**	**Sitzung**	**Dauer**	**Aktionen**	**CLS0**	...	**CLS52**
$r_{k_1,j}$	T_{k_1}	S_{i_1}	t_{i_1}	$\|C_{i_1,k_1}\|$	$C_{i_1,k_1,0}$	...	$C_{i_1,k_1,52}$
$r_{k_2,j}$	T_{k_2}	S_{i_2}	t_{i_2}	$\|C_{i_2,k_2}\|$	$C_{i_2,k_2,0}$	...	$C_{i_2,k_2,52}$
...							

Die vorliegende Tabelle beschreibt die in die Analyse der Rolle j eingehenden Daten unter Verwendung der zuvor eingeführten Definitionen. Jede Zeile enthält für einen Teilnehmer T_k dessen Ausprägung $r_{k,j}$ hinsichtlich der betrachteten Rolle sowie dessen der Sitzung S_i entnommenen Daten: Dies umfasst die Situngsdauer t_i (in Minuten), die Anzahl der vom Teilnehmer in der Sitzung ausgelösten Aktionen als Maß für dessen Beteiligung $A_{i,k}$ und den entsprechenden Code-Vektor $C_{i,k}$ (Spalten CLS0 bis CLS52).

Die Tabelle 10.4 illustriert diese formale Definition anhand eines konkreten Beispiels: Stellvertretend für sämtliche der zehn VitaminL-Rollen wird ein kleiner Ausschnitt aus den tabellarisch aufbereiteten kumulierten Sitzungsdaten des Umsetzers wiedergegeben.

[3] Von 2 der insgesamt 28 Teilnehmer liegen keine Rollenprofile vor, so dass diese nicht weiter betrachtet werden können.

Tab. 10.4: Kumulierte Sitzungsdaten des Umsetzers (Auszug)

Ausprägung	**Teilnehmer**	**Sitzung**	**Dauer**	**Beteiligung**	**CLS0**	**CLS1**	**CLS2**	**CLS3**	**CLS4**
14	T_{31}	S_7	99	2188	0	1	4	0	1
14	T_{31}	S_{13}	112	1537	0	1	0	0	1
14	T_{31}	S_{21}	105	2142	0	0	18	1	0
16	T_6	S_1	67	1009	0	1	0	0	2
16	T_{30}	S_7	99	544	0	0	4	0	0
16	T_6	S_{12}	121	997	0	0	1	2	4
16	T_{30}	S_{13}	112	169	0	1	1	1	1
16	T_{30}	S_{21}	105	1025	0	0	0	0	3
16	T_6	S_{22}	105	2828	0	0	2	2	0
17	T_{11}	S_3	94	2140	0	0	0	0	1

Die entsprechende Darstellung der zugehörigen beteiligungsbezogenen Sitzungsdaten ist nachfolgender Tabelle zu entnehmen. Die Werte in den mit CLS_i betitelten Spalten ergeben sich aus den entsprechenden Feldern (bzw. deren Werten) der Tabelle 10.4, dividiert durch die jeweilige Beteiligung eines Teilnehmers T_j (Spalte *Beteiligung*).

Tab. 10.5: Beteiligungsbezogene Sitzungsdaten des Umsetzers (Auszug)

Ausprägung	**Teilnehmer**	**Sitzung**	**Dauer**	**Beteiligung**	**CLS0**	**CLS1**	**CLS2**	**CLS3**	**CLS4**
14	T_{31}	S_7	99	2188	0	1,0101	4,0404	0	1,0101
14	T_{31}	S_{13}	112	1537	0	0,89286	0	0	0,89286
14	T_{31}	S_{21}	105	2142	0	0	17,14286	0,95238	0
16	T_6	S_1	67	1009	0	1,49254	0	0	2,98507
16	T_{30}	S_7	99	544	0	0	4,0404	0	0
16	T_6	S_{12}	121	997	0	0	0,82645	1,65289	3,30579
16	T_{30}	S_{13}	112	169	0	0,89286	0,89286	0,89286	0,89286
16	T_{30}	S_{21}	105	1025	0	0	0	0	2,85714
16	T_6	S_{22}	105	2828	0	0	1,90476	1,90476	0
17	T_{11}	S_3	94	2140	0	0	0	0	1,06383

Den Abschluss bildet die tabellarische Darstellung der zu den vorigen Beispielen korrespondierenden zeitbezogenen Sitzungsdaten. Die zeitbezogenen Werte der Teilnehmer (Spalten CLS_i) werden aus den entsprechenden Werten der Tabelle 10.4 berechnet, wobei die zugehörige Sitzungsdauer (Spalte *Dauer*) als Divisor in die Berechnung eingeht.

Tab. 10.6: Zeitbezogene Sitzungsdaten des Umsetzers (Auszug)

Ausprägung	**Teilnehmer**	**Sitzung**	**Dauer**	**Beteiligung**	**CLS0**	**CLS1**	**CLS2**	**CLS3**	**CLS4**
14	T_{31}	S_7	99	2188	0	0,0457	0,18282	0	0,0457
14	T_{31}	S_{13}	112	1537	0	0,06506	0	0	0,06506
14	T_{31}	S_{21}	105	2142	0	0	0,84034	0,04669	0
16	T_6	S_1	67	1009	0	0,09911	0	0	0,19822
16	T_{30}	S_7	99	544	0	0	0,73529	0	0
16	T_6	S_{12}	121	997	0	0	0,1003	0,2006	0,4012
16	T_{30}	S_{13}	112	169	0	0,59172	0,59172	0,59172	0,59172
16	T_{30}	S_{21}	105	1025	0	0	0	0	0,29268
16	T_6	S_{22}	105	2828	0	0	0,07072	0,07072	0
17	T_{11}	S_3	94	2140	0	0	0	0	0,04673

10.1.1.3.3 Filterung nicht-verwendeter Codes Bei näherer Betrachtung der in den Tabellen 10.4, 10.5 und 10.6 exemplarisch wiedergegebenen Daten fällt auf, dass ein hoher Anteil von Zellen mit dem Wert 0 belegt ist: Tatsächlich sind von den insgesamt 3.710 Code-Daten, die jede der betrachteten Tabellen beinhaltet, lediglich 1.396 Werte größer als 0, das heißt 2.314 Einträge sind gleich 0. Dies entspricht immerhin einem Anteil von 62,37 %, was wiederum bedeutet, dass – global betrachtet – von den Teilnehmern in den Sitzungen fast 2/3 der verfügbaren Aktionen nicht verwendet wurden.

Um die Analyseergebnisse von möglichen Störeffekten zu bereinigen, die sich aus der Berücksichtigung nicht-ausgelöster Aktionen ergeben können, werden sowohl die kumulierte Sitzungsdaten als auch die daraus gebildeten zeit- und beteiligungsbezogenen Sitzungsdaten in je einer weiteren Variante zur Verfügung gestellt, bei welcher alle mit 0 belegten Einträge entfallen. Diese Varianten werden nachfolgend als *bereinigte (kumulierte, zeit- oder beteiligungsbezogene) Sitzungsdaten* bezeichnet.

Die resultierende tabellarische Darstellung enthält folglich ausschließlich Daten derjenigen Codes, die von den Teilnehmern während einer Sitzung auch tatsächlich verwendet wurden. Unbenutzte Codes werden nun durch leere Tabellenzellen

repräsentiert. Beispielsweise ergibt sich die Tabelle 10.7 durch Bereinigung aus der Tabelle 10.6.

Tab. 10.7: Bereinigte zeitbezogene Sitzungsdaten des Umsetzers (Auszug)

Ausprägung	**Teilnehmer**	**Sitzung**	**Dauer**	**Beteiligung**	**CLS0**	**CLS1**	**CLS2**	**CLS3**	**CLS4**
14	T_{31}	S_7	99	2188		0,0457	0,18282		0,0457
14	T_{31}	S_{13}	112	1537		0,06506			0,06506
14	T_{31}	S_{21}	105	2142			0,84034	0,04669	
16	T_6	S_1	67	1009		0,09911			0,19822
16	T_{30}	S_7	99	544			0,73529		
16	T_6	S_{12}	121	997			0,1003	0,2006	0,4012
16	T_{30}	S_{13}	112	169		0,59172	0,59172	0,59172	0,59172
16	T_{30}	S_{21}	105	1025					0,29268
16	T_6	S_{22}	105	2828			0,07072	0,07072	
17	T_{11}	S_3	94	2140					0,04673

Insgesamt ergeben sich auf diese Weise sechs unterschiedliche Arten von Sitzungsdaten, die für jede der zehn VitaminL-Rollen in entsprechenden Tabellen bereitgestellt und in der Analyse verarbeitet werden.

10.1.1.3.4 Vereinfachende Notationsvereinbarung Zur Vereinfachung nachfolgender formaler Beschreibungen werden den diversen Code-Vektoren durch Hinzufügen eines zusätzlichen Index' alternative Benennungen wie folgt zugeordnet:

(a) Kumulierter Code-Vektor $C_{i,k} =: C^a_{i,k}$

(b) Bereinigter kumulierter Code-Vektor $\bar{C}_{i,k} =: C^b_{i,k}$

(c) Beteiligungsbezogener Code-Vektor $C^B_{i,k} =: C^c_{i,k}$

(d) Bereinigter beteiligungsbezogener Code-Vektor $\bar{C}^B_{i,k} =: C^d_{i,k}$

(e) Zeitbezogener Code-Vektor $C^T_{i,k} =: C^e_{i,k}$

(f) Bereinigter zeitbezogener Code-Vektor $\bar{C}^T_{i,k} =: C^f_{i,k}$

Die Anwendung dieser Numerierung lässt sich auch auf die jeweiligen Code-Kennzahlen anwenden: So lassen sich beispielsweise fortan die bereinigten kumulierten Code-Kennzahlen $\bar{C}^B_{i,k,l}$ auch als $C^b_{i,k,l}$ notieren, die zeitbezogenen

Code-Kennzahlen $C^b_{i,k,l}$ können als $C^b_{i,k,l}$ notiert werden. Für die übrigen Code-Kennzahlen gilt Entsprechendes.

10.1.2 Anwendung statistischer Verfahren

Formal geht es in der eingangs skizzierten Analyse darum, zwei Merkmale X und Y auf lineare Zusammenhänge hin zu untersuchen:

- Das erste Merkmal X stellt die Rollenausprägung $p_{k,n}$ eines Teilnehmers T_k bezüglich der Rolle R_n dar.
- Als zweites Merkmal Y gelangt eine Code-Kennzahl $C^x_{i,k,l}$ (mit $x \in \{a, b, c, d, e, f\}$) als quantitative Aussage über die Verwendung eines Codes $c_l \in CLS++$ zur Anwendung.

Dazu sind der Datenbasis geeignete Stichproben zu entnehmen.

10.1.2.1 Beschreibung der verwendeten Stichproben

Im Allgemeinen besteht eine Stichprobe (x, y) vom Umfang N aus N Beobachtungspaaren (x_1, y_1),(x_2, y_2),...,(x_N, y_N), wobei $(x_i, y_i) \in X \times Y$ für $1 \leq i \leq N$ gilt. Um beispielsweise im Falle der kumulierten Sitzungsdaten denZusammenhang zwischen einer Rolle R_j (mit $1 \leq j \leq 10$) und eines CLS++-Codes C_l (mit $1 \leq l \leq 52$) zu analysieren, gewinnt man eine Stichprobe, indem man über alle kumulierten Code-Vektoren $C_{i,k}$ iteriert und dabei ein Stichprobenelement (x, y) bildet mit $x = p_{k,j}$ und $y = C_{i,k,l}$. Anders formuliert setzt sich jedes Element der Stichprobe zusammen aus der Rollenausprägung $p_{k,j}$ eines Teilnehmers T_k hinsichtlich der zu untersuchenden Rolle R_j und der kumulierten Code-Kennzahl $C_{i,k,l}$ (entspricht der absoluten Häufigkeit der Verwendung des Codes C_l) des Teilnehmers T_k in einer Sitzung S_i.

Um beispielsweise die Rolle des Umsetzers (R_4) und den Code C_2 auf einen möglichen Zusammenhang hin zu untersuchen, gewinnt man die entsprechende Stichprobe anhand der zur Umsetzer-Rolle gehörenden Tabelle aus den Spalten *Ausprägung* und *CLS2*. Zur Verdeutlichung sind die entsprechenden Werte in Tabelle 10.8 hervorgehoben.

Somit ergibt sich für dieses Beispiel die (nur auszugsweise wiedergegebene) Stichprobe *((14,4) , (14,0) , (14,18) , (16,0), (16,4), (16,1), (16,1), (16,0), (16,2), (17,0), ...)*, die der weiteren Untersuchung zugeführt wird. Für die zeit- und beteiligungsbezogenen Code-Vektoren wird analog verfahren. Der Umfang N aller Stichproben dieser drei Arten von Code-Vektoren beträgt übrigens jeweils $N = 70$ (dies entspricht der Anzahl Teilnahmen; s. Tab.10.1).

Tab. 10.8: Stichprobengewinnung aus einer Sitzungsdatentabelle (Beispiel)

Ausprägung	**Teilnehmer**	**Sitzung**	**Dauer**	**Beteiligung**	**CLS0**	**CLS1**	**CLS2**	**CLS3**	**CLS4**
14	T_{31}	S_7	99	2188	0	1	**4**	0	1
14	T_{31}	S_{13}	112	1537	0	1	**0**	0	1
14	T_{31}	S_{21}	105	2142	0	0	**18**	1	0
16	T_6	S_1	67	1009	0	1	**0**	0	2
16	T_{30}	S_7	99	544	0	0	**4**	0	0
16	T_6	S_{12}	121	997	0	0	**1**	2	4
16	T_{30}	S_{13}	112	169	0	1	**1**	1	1
16	T_{30}	S_{21}	105	1025	0	0	**0**	0	3
16	T_6	S_{22}	105	2828	0	0	**2**	2	0
17	T_{11}	S_3	94	2140	0	0	**0**	0	1

Bei den jeweiligen bereinigten Varianten werden diejenigen Stichprobenelemente (x, y) eliminiert, bei denen $y = 0$ gilt. Infolgedessen reduzieren sich bei diesen Varianten für gewöhnlich die Umfänge der Stichproben.

10.1.2.2 Bestimmung des linearen Zusammenhangs

Bei der Frage nach einem möglichen linearen Zusammenhang zweier Merkmale beziehungsweise von Stichproben zweier Merkmale ergibt sich die Frage nach geeigneten Maßzahlen, anhand derer die Stärke des linearen Zusammenhangs bewertet werden kann. Die deskriptive Statistik liefert gleich mehrere solcher Maßzahlen, die aufeinander aufbauen.

10.1.2.2.1 Die empirische Kovarianz cov_{xy} Die empirische Kovarianz (auch als Produkt-Moment bezeichnet) cov_{xy} stellt solch ein Maß dar und wird für eine Stichprobe $(x, y) = (x_1, y_1),(x_2, y_2),\ldots,(x_N, y_N)$ vom Umfang N wie folgt definiert (vgl. Clauss et al., 1995, S.69):

$$cov_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}), \tag{10.15}$$

wobei mit $\bar{x}$ und $\bar{y}$ die arithmetischen Mittelwerte von x und y bezeichnet werden, die definiert sind als

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \text{ und } \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i. \tag{10.16}$$

Durch Anwendung des Verschiebungssatzes (vgl. VOGEL, 1979, S.20) kann die Kovarianz auch wie folgt definiert werden :

$$cov_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i y_i - N\bar{x}\bar{y}). \tag{10.17}$$

Wie leicht zu sehen ist, gibt die Kovarianz zwar die Richtung des Zusammenhangs zwischen x und y an, lässt aber in dieser Form keine Aussage über dessen Grad zu, da das Ergebnis von den verwendeten Maßeinheiten der betrachteten Merkmale abhängt. Diesem Mangel begegnet man, indem man die Kovarianz normiert.

10.1.2.2.2 Der Maßkorrelationskoeffizient r Der Maßkorrelationskoeffizient r (nach Bravais/Pearson)[4] ergibt sich aus der Normierung der Kovarianz und *„misst die Stärke und die Richtung linearer Beziehungen zwischen zwei metrischen Merkmalen X und Y"*(VOGEL, 1979, S.46), die beide in Form einer Stichprobe $(x, y) = (x_1, y_1),(x_2, y_2),\ldots,(x_N, y_N)$ vorliegen. Dieser Koeffizient wird in der gängigen Literatur (für eine Stichprobe (x, y)) definiert als

$$r_{xy} = \frac{N cov_{xy}}{(N-1)s_x s_y} \tag{10.18}$$

$$= \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{(N-1)s_x s_y}. \tag{10.19}$$

Hierbei ist s_x die empirische Standardabweichung von x[5] mit

$$s_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{10.20}$$

$$= \sqrt{s_x^2}. \tag{10.21}$$

Es wird folglich die empirische Standardabweichung s_x als Quadratwurzel aus der empirischen Varianz s_x^2 definiert. Diese wiederum gilt als ein Streumaß, das heißt die Varianz ist ein Maß für die Abweichung der beobachteten x-Werte vom arithmetischen Mittel $\bar{x}$. Für die Varianz gilt:

$$s_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2) \tag{10.22}$$

$$= \frac{1}{N-1} SAQ_x, \tag{10.23}$$

wobei die Summe der quadrierten Abweichungen aller einzelnen Messwerte x_i vom arithmetischen Mittelwert $x - 1$ auch als SAQ_x bezeichnet wird:

$$SAQ_x = \sum_{i=1}^{N}(x_i - \bar{x})^2, \tag{10.24}$$

[4] Der Maßkorreklationskoeffizient wird im weiteren Verlauf dieser Arbeit auch schlicht als Korrelationskoeffizient bezeichnet, da eine Unterscheidung zu anderen Korrelationskoeffizienten nicht notwendig ist.

[5] Für y gelten analoge Vereinbarungen und Definitionen.

so dass sich die Varianz auch darstellen lässt als

$$s_x^2 = \frac{1}{N-1} SAQ_x. \tag{10.25}$$

Wie in (Clauss et al., 1995, S.49) erläutert lässt sich SAQ_x auch in der Form

$$SAQ_x = \sum_{i=1}^{N} x_i^2 - N\bar{x}^2 \tag{10.26}$$

$$= \sum_{i=1}^{N} x_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} x_i\right)^2 \tag{10.27}$$

verwenden, wobei für die empirische Varianz wiederum Definition 10.25 gilt.

Ergänzend zu obiger Definition 10.18 lässt sich der Korrelationskoeffizient r_{xy} auch nach folgender Rechenformel bestimmen (vgl. Clauss et al., 1995, S.74f):

$$r_{xy} = \frac{SAQ_{xy}}{\sqrt{SAQ_x \cdot SAQ_y}}, \tag{10.28}$$

wobei SAQ_{xy} gilt:

$$SAQ_{xy} = \sum_{i=1}^{N} x_i y_i - \frac{1}{N}\left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right). \tag{10.29}$$

Die Definition von SAQ_x ist der Formel 10.24 zu entnehmen, für SAQ_y gilt eine entsprechende Definition.

Durch Berechnung des Korrelationskoeffizienten r_{xy} wird zunächst der Grad des linearen Zusammenhangs zwischen den Merkmalen X und Y einer Stichprobe (x, y) gemessen, wobei prinzipiell gilt (vgl. Büning & Trenkler, 1979, S.236f):

1. $-1 \leq r_{xy} \leq 1$
2. $r_{xy} = 0 \iff X$ und Y sind unkorreliert
3. $r_{xy} = 1 \iff Y = cX + d$ gilt mit Wahrscheinlichkeit 1 ($c \neq 0, d$ sind Konstanten).

Somit liegt für $|r = 1|$ zwar ein linearer Zusammenhang für die Werte X und Y in der Form $Y = a + bX$ vor, bei welchem der Korrelationskoeffizient r_{xy} als Gradzahl für das Maß des Zusammenhangs zwischen X und Y interpretiert werden kann, aber *„dabei darf der Maßkorrelationskoeffizient nicht als Prozentwert im Sinne von 'Anteil' eines linearen Zusammenhangs aufgefasst werden"*(Clauss et al., 1995, S.75).

10.1.2.2.3 Das Bestimmtheitsmaß B Solch ein Prozentwert wird üblicherweise durch die Größe $B = r_{xy}^2$ definiert, die in der Literatur als *Bestimmtheitsmaß* oder auch *Determinationskoeefizient* bezeichnet wird (s. Clauss et al., 1995, S.75,S.85; Vogel, 1979, S.45). Allgemein misst B die Stärke linearer Zusammenhänge und ist maximal, wenn die Restsumme der Abweichungsquadrate $= 0$ ist, das heißt alle Punkte liegen auf der sogenannten *Regressionsgeraden* (vgl. Röhr et al., 1983, S.249).

Es ergibt sich jedoch allein aus der Berechnung dieser Koeffizienten noch keine Aussage darüber, ob der Zusammenhang signifikant oder eher zufällig ist. Um dies zu ermitteln, werden – je nach der Art der Abhängigkeit zwischen den betrachteten Merkmalen – weitere Verfahren eingesetzt: Sofern die beiden Merkmale X und Y voneinander unabhängig sind, kann mittels einer Korrelationsanalyse das Vorliegen respektive der Grad eines linearen Zusammenhangs untersucht werden, wohingegen die Regressionsanalyse davon ausgeht, dass eines der beiden Merkmale unabhängig und das andere eine davon abhängige Größe darstellt (vgl. Clauss et al., 1995, S.302).

10.1.2.3 Korrelationsanalyse

In Anlehnung an die von Clauss et al. (1995) vorgeschlagene Vorgehensweise zur Durchführung statistischer Tests (vgl. Clauss et al., 1995, Kap.4.1.1) wird auch die Korrelationsanalyse als ein bivariabler Anpassungstest durch Abarbeitung folgender sechs Schritte durchgeführt:

1. Formulierung der Hypothesen
2. Festlegung des Signifikanzniveaus
3. Angabe der Testgröße
4. Angabe des kritischen Bereichs
5. Berechnung der Testgröße
6. Auswertung des Testergebnisses

Insgesamt deckt sich diese Vorgehensweise mit anderen in der Literatur formulierten Testverfahren (vgl. Gaensslen & Schubö, 1976; Röhr et al., 1983; Rüger, 2002).

10.1.2.3.1 Formulierung der Hypothesen Unter der Annahme, dass kein signifikanter linearer Zusammenhang zwischen Rollenausprägung (X) und Code-Kennzahl (Y) besteht, wird die zu prüfende Nullhypothese H_0 definiert mit $H_0 : \varrho_{XY} = 0$. Anders formuliert besagt die Nullhypothese, dass es keinen statistischen Effekt zwischen den beiden Merkmalen gibt.

Dieser Nullhypothese wird eine Alternative H_1 (oder Gegenhypothese) entgegengestellt und als $H_1 : \varrho_{XY} \neq 0$ formuliert: Somit unterstellt die Alternative eine systematische, das heißt signifikante Abweichung von 0 und geht von der Existenz eines statistischen Zusammenhangs zwischen den beobachteten Merkmalen aus.

Mit einem Test für H_0 gegen H_1 soll eine Entscheidung zugunsten einer der beiden Aussagen A_0 oder A_1 getroffen werden (vgl. RÜGER, 2002, S.9):

A_0 Die Nullhypothese H_0 wird nicht abgelehnt, sondern angenommen.

A_1 Die Alternative H_1 wird durch die Beobachtung statistisch nachgewiesen und ist somit signifikant, so dass H_0 abzulehnen ist.

Anhand konkreter Stichproben ist – für jede Stichprobe einzeln – eine Entscheidung hinsichtlich der beiden Aussagen A_0 oder A_1zu treffen. Der Stichprobenraum wird dabei in einen *kritischen Bereich* K (auch *Ablehnbereich* genannt) und einen *Annahmebereich* $\bar{K}$ zerlegt, wobei Kund $\bar{K}$ zueinander komplementär sind. Eine Beobachtung $x \in K$ führt dann zu einer Entscheidung zugungsten der Aussage A_1, wohingegen eine Beobachtung $x \in \bar{K}$ in der Aussage A_0 resultiert.

10.1.2.3.2 Festlegung des Signifikanzniveaus In diesem Zusammenhang kann es durchaus zu Fehlentscheidungen kommen, die in der Statistik als *Fehler 1. Art* beziehungsweise als *Fehler 2. Art* bezeichnet werden. *„Unter dem Fehler erster Art versteht man das Testergebnis A*1 *(H_0 ablehnen), obwohl in Wirklichkeit H_0 richtig ist"* (RÜGER, 2002, S.11). Die Wahrscheinlichkeit für solch eine Fehlentscheidung wird auch als *Irrtumswahrscheinlichkeit* (oder Risiko 1. Art) benannt und mit α bezeichnet. Dementsprechend bezeichnet ein Fehler 2. Art die Beibehaltung der Nullhypothese, obwohl sie falsch ist. Die Wahrscheinlichkeit dafür heißt auch Risiko 2. Art und wird mit β bezeichnet.

Tab. 10.9: Fehler 1. und 2. Art

		Testergebnis: Die Beobachtung führt zu der	
		Aussage A_0 (H_0 beibehalten)	Aussage A_1 (H_1 ist signifikant)
Realität	H_0 trifft zu	Richtige Aussage	Fehler 1. Art
	H_1 trifft zu	Fehler 2. Art	Richtige Aussage

Da α und β üblicherweise voneinander abhängen, hat die Wahl eines größeren α automatisch ein kleineres β zur Folge und umgekehrt. Aufgrund dieser Zusammenhänge beschränkt man sich üblicherweise auf die Festlegung von α und erhält somit einen *Signifikanztest zum Signifikanzniveau* α. Unter Berücksichtigung der Zielsetzung, signifikante lineare Zusammenhänge nachzuweisen und somit die Alternative H_1 zu bestätigen, ist das α so zu wählen, dass die Nullhypothese H_0 möglichst abgelehnt wird. Es wird $\alpha = 0,01$ festgelegt, wodurch die Nullhypothese

korrekterweise mit einer Wahrscheinlichkeit von $1 - \alpha = 0,99$ abgelehnt wird. Damit wird ein Risiko von 1 % akzeptiert, dass ein Fehler 1. Art auftritt.

10.1.2.3.3 Angabe und Berechnung der Testgröße Als Test- oder Prüfgröße kann direkt der Maßkorrelationskoeffizient r_{xy} gewählt werden (vgl. Clauss et al., 1995, S.299 unten). Dieser wird wie in Formel 10.28 festelegt durch

$$r_{xy} = \frac{SAQ_{xy}}{\sqrt{SAQ_x \cdot SAQ_y}} \tag{10.30}$$

berechnet.

10.1.2.3.4 Festlegung des kritischen Bereichs Da die Nullhypothese davon ausgeht, dass es keinen statistisch begründbaren Zusammenhang gibt, gilt: Wenn $|r| \geq r_{\alpha;m}$, dann ist H_0 abzulehnen. Die Vergleichswerte $r_{\alpha;m}$ für diverse α-Werte (5 %, 1 %, 0,27 % und 0,1 %) können der Tabelle D.1 im Anhang, Kapitel entnommen werden, wobei m der Freiheitsgrad ist, der sich aus dem Stichprobenumfang N ergibt durch $m = N - 2$.

10.1.2.3.5 Auswertung des Testergebnisses Den Abschluss der Korrelationsanalyse bildet die Durchführung des Vergleichs zwischen der errechneten Testgröße r_{xy} und des für die jeweilige Stichprobe (x, y) zulässigen Höchstwertes $r_{\alpha;m}$ (bei gegebenem Signifikanzniveau $\alpha = 0,01$ und Freiheitsgrad $m = N - 2$) mit anschließender Interpretation des Ergebnisses.

10.1.2.4 Regressionsanalyse

Im Unterschied zur Korrelationsanalyse geht man bei der Regressionsanalyse von der Annahme aus, dass eines der Merkmale X beziehungsweise Y unabhängig ist und das andere Merkmal davon abhängig ist. Die aus der graphischen Darstellung einer Stichprobe resultierende Punktwolke wird interpretiert als Ergebnis einer (hier: linearen) Funktion, auf die zufällige Störungen einwirken. Entsprechend des angenommenen Störeinflusses existieren zwei verschiedene Modelle zur linearen Regression. Das hier zur Verwendung kommende Modell II geht davon aus, dass sowohl die y- als auch die x-Werte Störeinflüssen unterliegen können. Entsprechend dieser Annahme werden nachfolgend Regressionsanalysen für beide Sichtweisen durchgeführt und sowohl Funktionen der Form $y = f(x)$ als $x = f(y)$ berechnet.

Während die im vorigen Abschnitt eingeführte Korrelationsanalyse Auskunft darüber erteilt, ob und inwiefern ein signifikaner beziehungsweise statistisch nachweisbarer Zusammenhang zwischen den Merkmalen X und Y existiert, versucht die hier durchgeführte Regressionsanalyse einen linearen Zusammenhang zwischen den beiden Merkmalen in Form einer Regressionsgeraden zu ermitteln.

10.1.2.4.1 Y in Abhängigkeit von X Die Regression von Y aus X behandelt die Schätzung von Y auf der Grundlage von X, wobei man von einer linearen Funktion der Form $y = a + bx$ ausgeht. Unter Berücksichtigung der Störungseinflüsse formuliert man auch genauer $y_i = a + bx_i + e_i$. Hierbei sind die e_i die jeweiligen Störungen, die sogenannten *Residuen*. Üblicherweise versucht man eine Näherungslösung in Form einer Geraden durch die Punktwolke zu finden, bei welcher die Quadratsumme der Residuen minimal ist. Ein übliches Verfahren stellt die Methode der kleinsten Quadrate dar (vgl. VOGEL, 1979, S.42,S.58ff), welches als gesuchte Näherungslösung die Funktion

$$\hat{y} = f(x) = \hat{a}_{yx} + \hat{b}_{yx}x \tag{10.31}$$

mit

$$\hat{b}_{yx} = \frac{SAQ_{xy}}{SAQ_x} \tag{10.32}$$

und

$$\hat{a}_{yx} = \bar{y} - \hat{b}_{yx}\bar{x} \tag{10.33}$$

liefert. Für die Störungen gilt hierbei $e_i = y_i - \hat{y}$.

10.1.2.4.2 X in Abhängigkeit von Y Die Regression von X aus Y erfolgt analog zu der eben behandelten Regression von Y aus X mit entsprechend geänderten Indizes. Somit ergibt sich die geschätzte lineare Funktion

$$\hat{x} = f(y) = \hat{a}_{xy} + \hat{b}_{xy}y \tag{10.34}$$

mit

$$\hat{b}_{xy} = \frac{SAQ_{xy}}{SAQ_y} \tag{10.35}$$

und

$$\hat{a}_{xy} = \bar{x} - \hat{b}_{xy}\bar{y}. \tag{10.36}$$

Diese zweite Näherungsfunktion wird lediglich aus Gründen der Vollständigkeit angegeben, weitere Untersuchungen und Berechnungen werden – sofern nicht anders angegeben – in der Regel auf der Grundlage von $f(x)$ erfolgen, zumal der lineare Zusammenhang in den Wertepaaren (x_i, y_i) unabhängig von der Betrachtungsweise (x abhängig von y oder y abhängig von x) existiert. Dies äußert sich auch letztlich im Bestimmtheitsmaß $B = r_{xy}^2$: Quadriert man die Formel 10.28 und setzt dort die Definitionen von $\hat{b}_{yx}$ (Formel 10.32) und $\hat{b}_{xy}$ (Formel 10.35), so erhält man

$$B = \hat{b}_{yx}\hat{b}_{xy}. \tag{10.37}$$

Nachfolgend werden daher ausschließlich die Koeffizienten $\hat{a}_{yx}$ und $\hat{b}_{yx}$ in weiteren Betrachtungen verwendet und zugunsten besserer Lesbarkeit als a und b notiert.

10.1.2.5 Ergänzende Notationsvereinbarungen

In Fortführung der in Kapitel 10.1.1.3.4 eingeführten Notationsvereinfachung durch Hizufügen eines weiteren Index' gilt auch für Kennzahlen wie den Maßkorrelationskoeffizienten r oder das Bestimmtheitsmaß B: Sofern eine Unterscheidung hinsichtlich der jeweils betrachteten Sitzungsdaten oder zugehöriger Code-Kennzahlen als notwendig oder sinnvoll erachtet wird, geschieht dies durch einen hochgestellten Index (aus der Menge $\{a, \ldots, f\}$) wie in Kapitel 10.1.1.3.4 vereinbart. Desweiteren werden die diversen Sitzungsdaten differenziert nach Rollen und CLS-Codes untersucht, so dass auch diese bei Bedarf in der Notation durch Berücksichtigung finden müssen. Auch dies erfolgt durch Einführung weiterer Indizes: Eine beliebige Variable v, die im Kontext einer Rolle R_n und eines CLS-Codes c_l betrachtet wird, kann im Bedarfsfall auch als $v_{n,l}$ notiert werden.

Ein Bestimmtheitsmaß B, das sich bei Betrachtung einer Rolle R_{n_1} und eines CLS-Codes c_{l_2} auf der Grundlage der kumulierten Sitzungsdaten ergibt, kann somit auch mit $B^a_{n_1,l_2}$ angegeben werden; die Koeffizienten a und b (genauer: $\hat{a}_{yx}$ und $\hat{b}_{yx}$), die sich aus der Regressionsanalyse der bereinigten zeitbezogenen Sitzungsdaten für Rolle R_{n_3} und einen CLS-Code c_{l_4} ergeben, werden entsprechend als $a^f_{n_3,l_4}$ und $b^f_{n_3,l_4}$ notiert.

10.2 Resultate der Rollenanalyse

Eine vollständige Darstellung sämtlicher Ergebnisse, die sich aus der Anwendung der Rollenanalyse auf die in Kapitel 10.1.1 erläuterten Sitzungsdaten in Kombination mit den zu den Teilnehmern gehörenden Rollenprofilen ergeben, lässt sich allein aufgrund des zu erwartenden Umfangs nicht sinnvoll gestalten. Stattdessen werden nachfolgend zunächst ausgewählte Einzelergebnisse präsentiert und diskutiert.

10.2.1 Analyseresultate der Rolle *Informationsbeschaffer*

Die Untersuchung des Informationsbeschaffers als wichtigste Rolle für die Lernsituation ist insofern bedeutsam, da einer Umsetzung verwertbarer Ergebnisse eine hohe Relevanz für die tutorielle Unterstützung beigemessen wird.

10.2.1.1 Ergebnisse der kumulierten Sitzungsdaten

Beginnend mit der Betrachtung aller erhobenen und durch Kumulation verdichteten Sitzungsdaten lässt sich am Beispiel des CLS++-Codes 0 (Kommunikationsattribut *Koordination*: *„Okay, lasst uns fortfahren."*) die Analyse stellvertretend für die restlichen Codes durchführen. Aus der Stichprobe vom Umfang $N = 70$[6] wird

[6] Die Wiedergabe sämtlicher 70 Wertepaare mag an dieser Stelle aus Gründen der Übersichtlichkeit verzichtet werden.

zunächst der Maßkorrelationskoeffizient $r_{xy} = 0,3427$ berechnet. Aus diesem ergibt sich das Bestimmtheitsmaß $B = r_{xy}^2 = 0,1175$, das heißt 11,75 % der Varianz von Code 0 können durch die Ausprägung des Informationsbeschaffers erklärt werden.

Um nun zu überprüfen, ob das Ergebnis signifikant ist und somit ein statistisch nachweisbarer Zusammenhang zwischen Rollenausprägung und der Verwendung von Code 0 besteht, wird der Betrag des Maßkorrelationskoeffizienten mit dem kritischen Bereich (zum zuvor vereinbarten Signifikanzniveau $\alpha = 0,01$ mit Freiheitsgrad $m = N - 2 = 68$) verglichen: $|r_{xy}| = 0,3427 \geq 0,3042 = r_{\alpha;m}$. Somit ist die Nullhypothese H_0 abzulehnen, die ja fomuliert, dass kein solcher Zusammenhang besteht, es gilt folglich die Alternative H_1 und damit die zugehörige Aussage A_1 (*„Es gibt einen statistisch nachweisbaren Zusammenhang zwischen den Beobachtungen."*).

Die anschließende Regressionsanalyse liefert als Ergebnis die Funktionen $\hat{y} = f(x) = \hat{a}_{yx} + \hat{b}_{yx}x = -0,9068 + 0,0587x$ und $\hat{x} = f(y) = \hat{a}_{xy} + \hat{b}_{xy}y = 19,1137 + 2,0025y$ als Näherungen. Die nachfolgende Abbildung 10.2 enthält sowohl die verwendete Stichprobe (in Form einer Punktwolke) sowie die resultierende Näherungsfunktion $y = f(x)$.

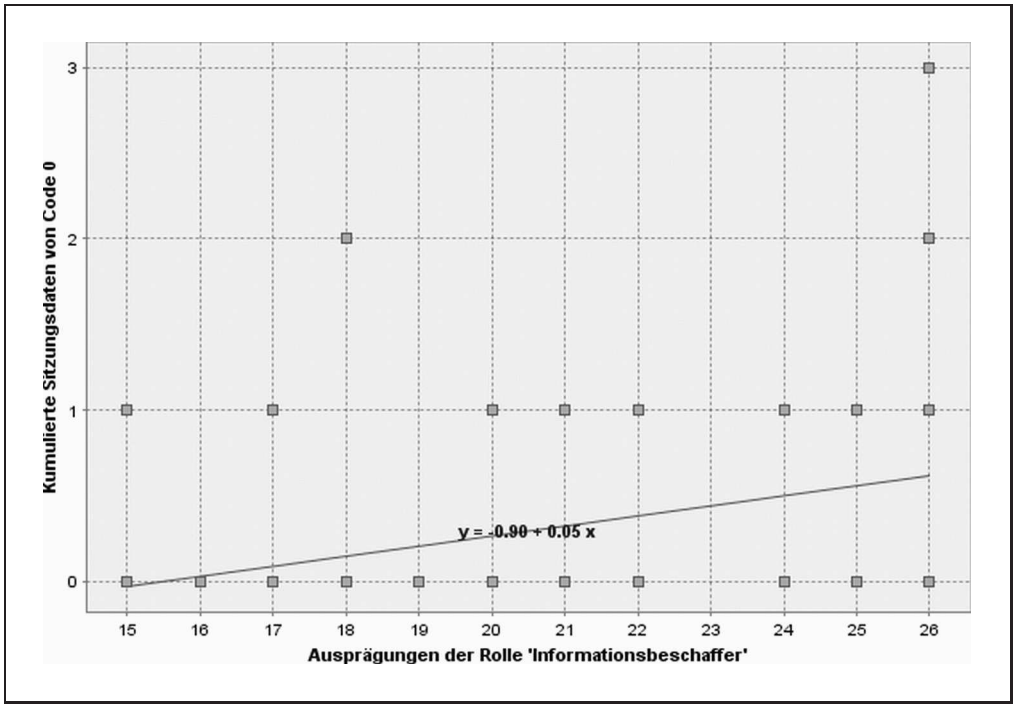

Abb. 10.2: Analyseergebnis: Rolle *IB*, Code 0, Variante (a)

Wie deutlich zu sehen ist, stellt die errechnete Funktion $y = f(x)$ nur eine sehr schwache Annäherung an die beobachtete Stichprobe dar. Dies wird auch durch den Wert des Bestimmtheitsmaßes $B = 0,1175$ bestätigt, welches ja einen lediglich knapp 12 %-igen linearen Zusammenhang zwischen den x- und y-Werten erklärt. Für die Rollenanalyse bedeutet dies, dass es keinen nenneswerten signifikanten Zusammenhang zwischen der Rolle des Informationsbeschaffers und der Verwendung von CLS++-Code 0 gibt. Anders formuliert: Allein aus der Verwendung von

CLS++-Code 0 lässt sich nicht auf die Ausprägung eines Teilnehmers hinsichtlich der Rolle des Informationsbeschaffers schließen.

Die Untersuchung der kumulierten Sitzungsdaten für den Informationsbeschaffer resultiert in drei weiteren CLS++-Codes, die einen signifikanten linearen Zusammenhang besitzen: Code 28 (Kommunikationsattribut *Ausarbeitung* mit $B = 0,0943$), Code 29 (Kommunikationsattribut *Erklärung* mit $B = 0,1188$) und Code 30 (Kommunikationsattribut *Rechtfertigung* mit $B = 0,1496$). Zwar führen auch diese Codes zu einer Annahme der Hypothese H_1, jedoch ist der lineare Zusammenhang zwischen den beobachteten Merkmalen mit jeweils weniger als 15 % insgesamt nur schwach ausgeprägt. Alle anderen Codes weisen keinen statistisch nachweisbaren Zusammenhang auf.

10.2.1.2 Ergebnisse der weiteren Sitzungsdaten

Die Untersuchung der übrigen Sitzungsdaten liefert für die Rolle des Informationsbeschaffers folgende Ergebnisse:

- Die bereinigten kumulierten Sitzungsdaten des Informationsbeschaffers besitzen nur für den Code 24 (Kommunikationsattribut *Zweifel*: *„Ich bin nicht sicher..."*) einen nachweisbaren linearen Zusammenhang bei einem Bestimmtheitsmaß von $B = 0,1843$ und einem Stichprobenumfang von $N = 39$.
- Auch innerhalb der beteiligungsbezogenen Sitzungsdaten kann nur für einen einzigen Code ein linearer Zusammenhang zwischen den beobachteten Merkmalen nachgewiesen werden: Code 41 (Dateioperation *Speichern*) ergibt ein Bestimmtheitsmaß von $B = 0,0939$ ($N = 70$).
- Die Untersuchung der bereinigten beteiligungsbezogenen Sitzungsdaten führt für keinen einzigen der 53 CLS++-Codes zu einem statistisch nachweisbaren Zusammenhang, das heißt es wurde jeweils die Nullhypothese angenommen.
- Die Ergebnisse hinsichtlich der zeitbezogenen Sitzungsdaten ähneln denen der kumulierten Sitzungsdaten: Für die Codes 0, 28, 29 und 30 können lineare Zusammenhänge mit den Bestimmtheitsmaßen 0,1144 (Code 0), 0,0936 (Code 28), 0,1314 (Code 28) und 0,1552 (Code 30) statistisch nachgewiesen werden. Es gilt jeweils $N = 70$ für den Stichprobenumfang.
- Bei Betrachtung der bereinigten zeitbezogenen Sitzungsdaten kann lediglich für den Code 11 (Kommunikationsattribut *Information*: *„Weisst Du...?"*) ein Zusammenhang mit einem Bestimmtheitsmaß $B = 0,1787$ aufgedeckt werden. Die verwendete Stichprobe beinhaltet $N = 41$ Elemente.

Insgesamt beinhalten die sechs Arten von Sitzungsdaten für die Rolle des Informationsbeschaffers sieben Codes mit elf statistisch nachweisbaren linearen Zusammenhängen, jedoch liegen sämtliche Bestimmtheitsmaße deutlich unter 0,2. Somit können in jedem Fall weniger als 20 % der Varianz eines Codes durch den Informationsbeschaffer erklärt werden.

10.2.2 Analyseresultate der Rolle *Berater*

Die Untersuchung der Sitzungsdaten mit Blick auf die Rollenausprägungen des Beraters ergeben zunächst ähnliche Resultate wie beim zuvor betrachteten Informationsbeschaffer: Innerhalb der kumulierten Sitzungsdaten lassen sich nur für drei Codes lineare Zusammenhänge nachweisen, deren Bestimmtheitsmaße allesamt unter 0,18 liegen. Konkret sind dies der Code 7 (Kommunikationsattribut *Zuhören, Verstehen*) mit $B = 0,1303$, der Code 21 (Kommunikationsattribut *Folgerung*) mit $B = 0,1784$ und der Code 24 (Kommunikationsattribut *Annahme*) mit $B = 0,1766$. Dieselben drei Codes ergeben sich auch bei der Untersuchung der zeitbezogenen Sitzungsdaten, weisen dort jedoch noch geringere Bestimmtheitsmaße auf: Code 7 mit $B = 0,1153$, Code 21 mit $B = 0,1284$ und Code 24 mit $B = 0,1048$. Die beteiligungsbezogenen Sitzungsdaten liefern statistische Zusammenhänge für die Codes 10 (Kommunikationsattribut *Aufmerksamkeit erbitten*, $B = 0,1028$), 19 (Kommunikationsattribut *Widerspruch*, $B = 0,0941$) und 21 (Kommunikationsattribut *Folgerung*, $B = 0,1750$). Interessanterweise taucht Code 21 bei allen drei Kategorien unbereinigter Sitzungsdaten auf – und liefert dort jeweils das höchste Bestimmtheitsmaß –, jedoch sind die Ergebnisse (in Form der Bestimmtheitsmaße) insgesamt eher gering, so dass ein im Hinblick auf eine Umsetzung innherhalb der Tutorkomponente verwertbarer linearer Zusammenhang kaum ableitbar ist.

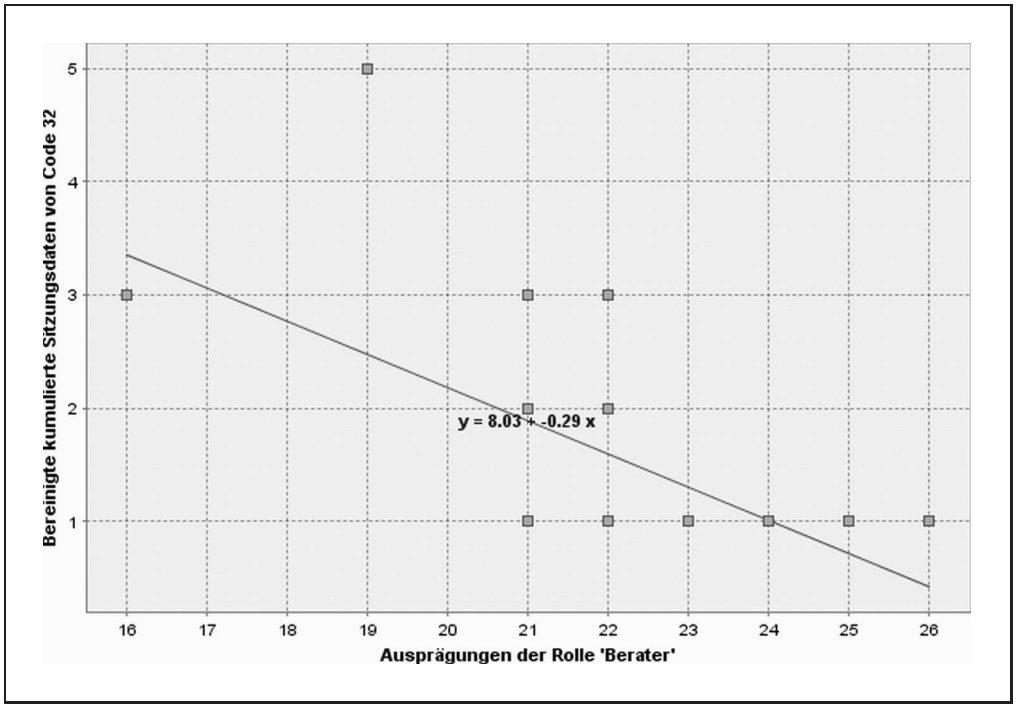

Abb. 10.3: Analyseergebnis: Rolle *BR*, Code 32, Variante (b)

Bei Betrachtung der bereinigten Sitzungsdaten des Beraters ergeben sich wesentlich andere Resultate. Innerhalb der bereinigten kumulierten Sitzungsdaten lassen sich bei zwei CLS-Codes lineare Zusammenhänge statistisch nachweisen: Die Analyse von Code 24 (Kommunikationsattribut *Annahme*) liefert ein Bestimmtheitsmaß

$B = 0,1630$ bei einem Stichprobenumfang von $N = 39$ und liegt somit im Trend bisheriger Ergebnisse. Bei der Untersuchung von Code 32 (Kommunikationselement *Ermutigung*) ergibt sich immerhin ein Bestimmtheitsmaß $B = 0,4122$ für die Stichprobe vom Umfang $N = 20$, das heißt es lassen sich für diesen Code bereits über 40 % der Stichprobenvarianz erklären. Die Stichprobe sowie die resultierende Näherungsfunktion sind in der Abbildung 10.3 wiedergegeben.

Die Untersuchung der bereinigten beteiligungsbezogenen Sitzungsdaten ergibt für den Berater keine signifikanten Ergebnisse: Keiner der untersuchten CLS++-Codes führt zur Ablehnung der Nullhypothese H_0. Im Gegensatz dazu liefert die Analyse der bereinigten zeitbezogenen Sitzungsdaten zwei weitere Codes, bei denen sich ein linearer Zusammenhang zwischen Rollenausprägung und Sitzungsdaten nachweisen lässt: Sowohl für Code 22 (Kommunikationselement *Annahme*; $B = 0,3461$ und $N = 19$) als auch für Code 32 (Kommunikationselement *Ermutigung*; $B = 0,3063$ und $N = 13$) lassen sich jeweils über 30 % der Abweichung vom Mittelwert der Stichprobe erklären.

10.2.3 Analyseresultate der Rolle *Problemlöser*

Der Problemlöser stellt eine interessante Rolle in dem Lernszenario dar: Aus Sicht der Teilnehmer wird dieser Rolle eine sehr hohe Bedeutung zugemessen (s. Kap.9.3.3.2), wohingegen sie aufgrund ihrer Relevanz für ein tutorielles System im VitaminL-Rollenmodell eher im Mittelfeld anzusiedeln ist (s. Kap.9.3.3.1). Die Untersuchung der Sitzungsdaten im Hinblick auf mögliche Zusammenhänge zwischen verwendeten CLS++-Codes und Ausprägungen der Problemlöser-Rolle soll daher die Reihe der präsentierten Einzelergebnisse schließen.

Betrachtet man zunächst die Analyseergebnisse der kumulierten Sitzungsdaten, so offenbaren diese wenig Neues: Bei insgesamt drei Codes (7, 24 und 27) kann ein Zusammenhang statistisch begründet werden, jedoch liegen sämtliche Bestimmtheitsmaße als eine Art Gütekriterium deutlich unter 20 % (Code 7 mit $B = 0,1787$, Code 24 mit $B = 0,1005$ und Code 27 mit $B = 0,0938$). Wie bei den zuvor untersuchten Rollen finden sich sehr ähnliche Ergebnisse in den zeitbezogenen Sitzungsdaten wieder: Für Code 7 ergibt sich ein Bestimmtheitsmaß von $B = 0,1684$, das Bestimmtheitsmaß für Code 27 beträgt $B = 0,1037$. Die Analyse der beteiligungsbezogenen Sitzungsdaten liefert keine signifikanten Zusammenhänge.

Die Untersuchung der bereinigten kumulierten Sitzungsdaten liefert für vire Codes einen signifikanten Zusammenhang, von denen einer aus allen bisherigen Ergebnissen deutlich hervorsticht: Die Stichprobe für Code 30 (vom Umfang $N = 7$) resultiert in einer Bestätigung der Hypothese H_1 bei einem Bestimmtheitsmaß von $B = 0,8428$. Die resultierende Näherungsfunktion sowie die zugehörige Stichprobe können der folgenden Abbildung 10.4 entnommen werden. Die übrigen Ergebnisse bewegen sich hingegen im üblichen Rahmen: Für Code 11 ergibt sich $B = 0,2931$ (Umfang der Stichprobe: $N = 41$), Code 24 liefert $B = 0,2193$ ($N = 39$) und Code 27 hat als Ergebnis $B = 0,1324$ ($N = 58$).

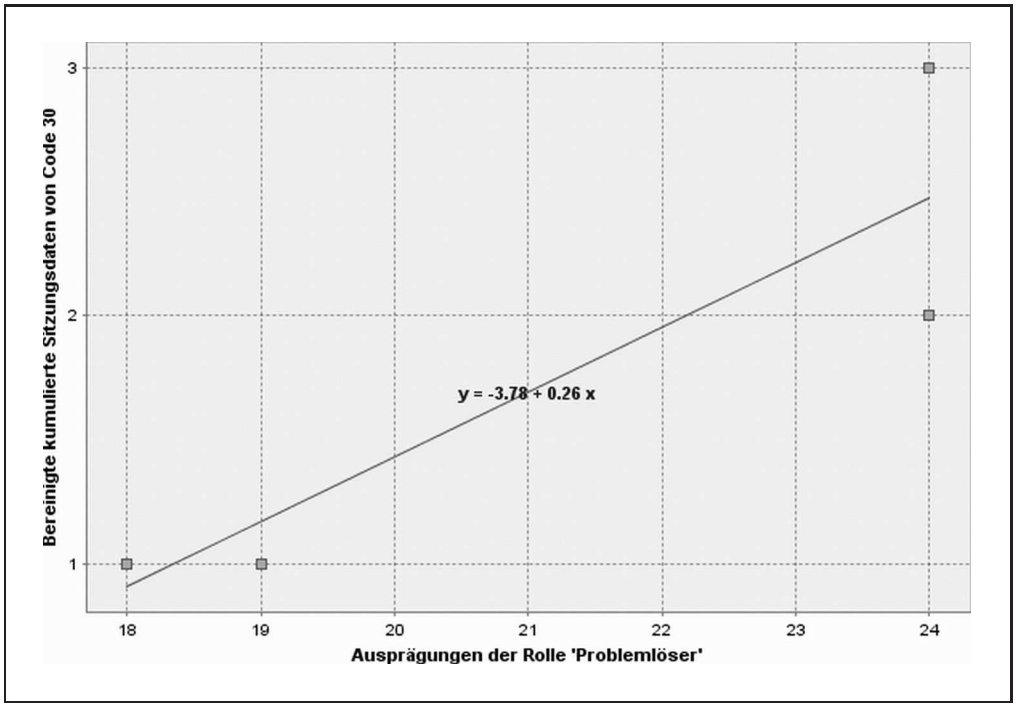

Abb. 10.4: Analyseergebnis: Rolle *PB*, Code 30, Variante (b)

Wie in der Abbildung zu sehen ist, erfährt die Stichprobe von Code 30 durch die Regressionsgerade $y = -3,78 + 0,26x$ eine gute Näherung, was sich auch in dem vergleichsweise hohen Bestimmtheitsmaß ausdrückt. Im Gegensatz zu den meisten bislang diskutierten Einzelergebnissen ist jedoch der Umfang der zugrundeliegenden Stichprobe mit $N = 7$ vergleichsweise gering. Insofern ist das Ergebnis für Code 30 trotz des hohen linearen Zusammenhangs und der daraus resultierenden Näherung kritisch zu betrachten und auf jeden Fall einer Überprüfung mittels weiterer Sitzungen zu unterziehen.

Die bereinigten beteiligungsbezogenen Sitzungsdaten für den Problemlöser enthalten keine signifikanten Ergebnisse, hingegen liefern die bereinigten zeitbezogenen Sitzungsdaten weitere diskussionswürdige Resultate: Code 30 liefert auch hier mit $B = 0,8424$ ($N = 7$) einen ähnlich hohen Wert wie bei den bereinigten kumulierten Sitzungsdaten, was eine Annahme der Hypothese für diesen Code mit sich führt. Alle anderen signifikanten Ergebnisse der Rolle *Problemlöser* besitzen nur vergleichsweise kleine Werte für B und werden daher nicht weiter betrachtet.

10.2.4 Zusammenfassung der Analyseergebnisse

Alle Codes, deren Untersuchungen signifikante Ergebnisse liefern, sind in der nachfolgenden Tabelle 10.10 zusammengefasst, geordnet nach Rollen (Zeilen) und verwendete Sitzungsdaten (Spalten). Somit enthält jedes Feld dieser Tabelle diejenigen Codes, die bei Betrachtung der jeweiligen Sitzungsdaten für die entsprechende Rolle statistisch nachweisbare Zusammenhänge ergeben. Zu jedem Ergebnis (in Form eines CLS++-Codes) wird das Bestimmtheitsmaß als Gütekriterium, das

heißt als Indikator für die Qualität des Ergebnisses angegeben.

Tab. 10.10: Signifikante Ergebnisse der Rollenanalyse (Sitzung S_1-S_{27})

Rolle	$C^a_{i,k}$	$C^b_{i,k}$	$C^c_{i,k}$	$C^d_{i,k}$	$C^e_{i,k}$	$C^f_{i,k}$
IB	0 : 0,1175 28 : 0,0943 29 : 0,1189 30 : 0,1496	24 : 0,1843	41 : 0,0939		0 : 0,1200 28 : 0,0937 29 : 0,1122 30 : 0,1509	24 : 0,1695
PL	2 : 0,1096 28 : 0,1139 39 : 0,1274	39 : 0,2499			2 : 0,1112 28 : 0,1121 30 : 0,0946	39 : 0,1680
BR	7 : 0,1303 21 : 0,1785 24 : 0,1767	24 : 0,1630 32 : 0,4123	10 : 0,1029 19 : 0,0941 21 : 0,1750		7 : 0,1298 21 : 0,1711 24 : 0,1593	
UM	4 : 0,1197 11 : 0,1532 20 : 0,0963 21 : 0,1339 24 : 0,0928 41 : 0,1079	4 : 0,1223 11 : 0,1786 41 : 0,1580		36 : 0,1469 41 : 0,1236	4 : 0,1119 11 : 0,1594 20 : 0,0952 21 : 0,1259 41 : 0,1411	11 : 0,1909 41 : 0,2147
SL	24 : 0,1136	47 : 0,3863	26 : 0,0958 33 : 0,0946 36 : 0,1009		24 : 0,0965	47 : 0,4324
PB	7 : 0,1787 24 : 0,1005 27 : 0,0939	11 : 0,2931 24 : 0,2193 27 : 0,1325 30 : 0,8428			7 : 0,1693 24 : 0,0936 27 : 0,1007	11 : 0,2707 24 : 0,1873 27 : 0,1431 30 : 0,8244
FR	11 : 0,0979 12 : 0,0954 28 : 0,1085	11 : 0,1601 28 : 0,3735 29 : 0,1960			12 : 0,0961 28 : 0,1076	28 : 0,3690 29 : 0,1986
MD	28 : 0,1105 39 : 0,1057	39 : 0,2039			28 : 0,1096	12 : 0,3482 39 : 0,1497
AR				29 : 0,2571		
VP	4 : 0,1176				4 : 0,1357	

Bei näherer Betrachtung der Tabelle fällt auf, dass bei mehreren Rollen einige CLS++-Kategorien (s. Tab.8.4) häufiger mit signifikanten Resultaten vertreten sind als andere. Der Informationsbeschaffer beispielsweise weist von seinen elf Ergebnissen immerhin sechs Ergebnisse aus der Kategorie *F. Information* (Codes 28, 29 und 30) auf. Dieses kann durchaus als Stützung der in Kapitel 7.3.4.2 formulierten These angesehen werden, wonach anhand des Verhaltens eines Teilnehmers auf dessen Rollenzugehörigkeit geschlossen werden kann. Weitere Indizien finden sich beim Fragesteller, dem vier von zehn Ergebnissen der Kategorie D. Anforde-

rung zuzuordnen sind: Code 11 (*„Weisst Du...?"*) und Code 12 (*„Kannst Du mir mehr sagen...?"*) liefern je zwei Ergebnisse und können für die in Kapitel 9.2.1.4 charakterisierte Rolle als typische Fragestellungen verstanden werden.

Insgesamt liefert die Durchführung der Rollenanalyse auf Basis der in Kapitel 10.1.1.1 beschriebenen Sitzungen und der dort erfassten Daten 88 lineare Zusammenhänge zwischen Rollenausprägungen und verwendeten Codes, die statistisch nachweisbar sind und geeignet umzusetzen sind, um die VitaminL-Software um eine Tutorkomponente mit Rollenerkennung zu erweitern. Die zugehörigen Ergebnisse der Regressionsanalyse finden sich im Anhang in den Tabellen D.2 bis D.7.

10.3 Evaluierung der Rollenanalyse

Die Überprüfung der Ergebnisse der Rollenananlyse erfolgt anhand weiterer Benutzertests, die im Dezember 2005 durchgeführt wurden. Bei diesen Sitzungen kam eine spezielle Version der VitaminL-Software zum Einsatz: Der VitaminL-Server wurde um einen Prototypen einer Komponente zur Rollenanalyse erweitert, welcher gemäß der in Abbildung 8.30 schematisch dargestellten Architektur zunächst sämtliche von den Teilnehmern ausgelösten Nachrichtenobjekte analysiert und in Rollenausprägungen umrechnet. Die auf diese Weise errechneten Ergebnisse wurden zunächst protokolliert und nach Abschluss der Testphase den Rollenprofilen der Teilnehmer gegenübergestellt, um aus diesem Vergleich Schlüsse hinsichtlich der Qualität der Ergebnisse sowie der einzelnen Analyseverfahren zu ziehen.

10.3.1 Berechnung von Rollenausprägungen

Ausgehend von den linearen Zusammenhängen zwischen Rollenausprägungen und CLS-Codes beziehungsweise den aus der Verwendung der CLS-Codes resultierenden Code-Kennzahlen, die im Rahmen der zuvor getätigten Rollenanalyse aufgedeckt werden konnten, ist ein Verfahren zu entwickeln und umzusetzen, mit dessen Hilfe aus den CLS-Codes, die von den Teilnehmern während einer Sitzung verwendet werden, auf die Rollenausprägungen der jeweiligen Teilnehmer geschlossen werden kann.

Den Ausgangspunkt für solche eine Berechnung bildet (für jedes Einzelergebnis der Rollenanalysen) die Regressionsgerade, über die ein linearer Zusammenhag zwischen einer Rollenausprägung p und einer Code-Kennzahl c in der Form

$$c = a + bp \tag{10.38}$$

definiert wird. Dabei sind a und b die per Regressionsanalyse errechnete Koeffizienten der jeweiligen Regressionsgeraden. Durch Umformen nach p erhält man

$$p = \frac{c - a}{b} \tag{10.39}$$

als Ansatz, um aus einer Code-Kennzahl c eine Rollenausprägung p zu berechnen.

Sämtliche Berechnungen basieren stets auf den Daten einer Sitzung S_i (mit einer Dauer t_i in Minuten) und werden durchgeführt

- für einen Teilnehmer T_k, der in der Sitzung S_i eine Beteiligung $A_{i,k}$ erreicht hat
- unter Verwendung der kumulierten, zeit- oder beteiligungsbezogenen Code-Kennzahlen ($C_{i,k,l}$, $C^T_{i,k,l}$ oder $C^B_{i,k,l}$) beziehungsweise einer der bereinigten Varianten ($\bar{C}_{i,k,l}$, $\bar{C}^T_{i,k,l}$ oder $\bar{C}^B_{i,k,l}$)
- für eine Rolle R_n
- bei Betrachtung eines CLS-Codes c_l
- mit den zugehörigen Koeffizienten a (bzw. $a^a_{n,l}$, $a^b_{n,l}$, $a^c_{n,l}$, $a^d_{n,l}$, $a^e_{n,l}$ oder $a^f_{n,l}$) und b (bzw. $b^a_{n,l}$, $b^b_{n,l}$, $b^c_{n,l}$, $b^d_{n,l}$, $b^e_{n,l}$ oder $b^f_{n,l}$), die aus der Regressionsanalyse von Rolle R_n und CLS-Code c_l mit entsprechenden Code-Kennzahlen resultieren.

10.3.1.1 Basis: Kumulierte Code-Kennzahlen

Die Berechnung einer Rollenausprägung $p^a_{i,k,n,l}$ auf der Basis der kumulierten Code-Kennzahlen $C^a_{i,k,l}$ eines Teilnehmers T_k innerhalb einer Sitzung S_i wird für eine Rolle R_n und einen CLS-Code c_l wie folgt durchgeführt:

$$p^a_{i,k,n,l} = \frac{C^a_{i,k,l} - a^a_{n,l}}{b^a_{n,l}}, \tag{10.40}$$

wobei $a^a_{n,l}$ und $b^a_{n,l}$ die Koeffizienten der Regressionsgeraden sind, die aus der zuvor durchgeführten Regressionsanalyse kumulierter Sitzungsdaten für Rolle R_n und CLS-Code c_l resultieren.

10.3.1.2 Basis: Beteiligungsbezogene Code-Kennzahlen

Die entsprechende Berechnung auf der Grundlage der beteiligungsbezogenen Code-Kennzahlen $C^c_{i,k,l}$ ist definiert durch:

$$p^c_{i,k,n,l} = \frac{C^c_{i,k,l} - a^c_{n,l}}{b^c_{n,l}} \tag{10.41}$$

$$= \frac{\frac{C^a_{i,k,l}}{A_{i,k}\frac{1}{100}} - a^c_{n,l}}{b^c_{n,l}}. \tag{10.42}$$

10.3.1.3 Basis: Zeitbezogene Code-Kennzahlen

Für die Berechnung mittels zeitbezogener Code-Kennzahlen gilt analog:

$$p^e_{i,k,n,l} = \frac{C^e_{i,k,l} - a^e_{n,l}}{b^e_{n,l}} \tag{10.43}$$

$$= \frac{\frac{C^a_{i,k,l}}{t_i \frac{1}{100}} - a^e_{n,l}}{b^e_{n,l}}. \tag{10.44}$$

Entsprechendes gilt für die jeweiligen Berechnungen auf Basis der bereinigten Code-Kennzahlen. Die für die Berechnungen benötigten Koeffizienten a und b sind in den Tabellen D.2 bis D.7 zusammengefasst.

10.3.2 Prototypische Umsetzung der Rollenanalyse

Im Rahmen einer sukzessiven Erweiterung der VitaminL-Software von einem CSCL-System zu einem ITS stellt die Entwicklung einer Komponente zur Durchführung der Rollenanalyse einen wichtigen Schritt dar. Der Prototyp solch einer Analysekomponente dient der Überprüfung der Analyseergebnisse, indem die im vorigen Kapitel skizzierten Berechnungen umgesetzt und in die bestehende Software integriert werden. Pro Teilnehmer einer Sitzung wird innerhalb der Tutorkomponente (vom Typ `VLernerModell`) eine Instanz der Klasse `VLernendenModell` generiert, die für den jeweiligen Teilnehmer dessen Rollenausprägungen anhand der eingehenden Nachrichten berechnet.

Beispiel 10.1: Analyse eines Nachrichtenobjekts in der Tutorkomponente

```
void VLernerModell::analyzeMessage( VMessage msg )
{
  // Hole Referenz auf Rollenanalysekomponente lm vom Typ VLernendenModell
  // des Teilnehmers, der die Nachricht msg ausgeloest hat: msg.getUser()
  VLernendenModell lm = getLernendenModell( msg.getUser() );

  // Uebergib das Nachrichtenobjekt msg an die Analysekomponente
  // zur weiteren Verarbeitung, d.h. Berechnung von Rollenauspraegungen
  lm.analyzeMessage( msg );

  // Informiere alle Interessenten ueber aktualisierte Rollenwerte
  updateAllListeners());
}
```

■

Da die Berechnung von Rollenausprägungen stets auf der kumulierten Code-Kennzahl (d.h. den absoluten Häufigkeiten der CLS-Codes) eines jeden Teilnehmers basiert (s. Kap.10.1.1.3.1), ist es notwendig, diese zunächst innerhalb der Analysekomponente vorrätig zu halten: Dies geschieht, indem pro CLS-Code ein Zähler (mit Anfangswert = 0) bereitgestellt wird, der bei jeder eintreffenden Nachricht

mit identischem CLS-Code um 1 erhöht wird. Ergänzend werden in einem weiteren Zähler sämtliche eingehenden Nachrichten des Benuters gezählt. Anschließend werden die Rollenausprägungen des betreffenden Benutzers, die vom eingehenden CLS-Code und den geänderten Zählerwerten abhängen, erneut berechnet. Dabei müssen – je nach Art der Code-Kennzahlen (kumuliert, beteiligungs- oder zeitbezogen) – auch die jeweils aktuelle Sitzungsdauer oder die Gesamtzahl der Aktionen berücksichtigt werden.

Beispiel 10.2: Prinzip der Rollenanalyse: Klasse `VLernendenModell`

```
void VLernendenModell::analyzeMessage( VMessage msg )
{
  // Lies aus der Nachricht deren CLS-Code aus
  int clsCode = msg.getCLSCode();

  // Erhoehe dessen absolute Haeufigkeit
  clsZaehler[clsCode]++;

  // Berechne alle zugehoerigen Rollenauspraegungen
  // auf Basis der kumulierten Sitzungsdaten
  berechneRollenSitzungsdatenKumuliert( clscode );

  // Erhoehe den Zaehler fuer die Beteiligung des Teilnehmers...
  clsZaehlerSumme++;
  // ...und berechne alle zugehoerigen Rollenauspraegungen
  // auf Basis der beteiligungsbezogenen Sitzungsdaten
  berechneRollenSitzungsdatenBeteiligung( clscode );

  // Berechne die aktuelle Sitzungsdauer in Minuten...
  int currentTimeInSecs = delta( msg.getTimeStamp() , startTimeStamp );
  int currentTimeInMins = currentTimeInSecs / 60;
  // ...und berechne alle zugehoerigen Rollenauspraegungen
  // auf Basis der zeitbezogenen Sitzungsdaten
  berechneRollenSitzungsdatenZeit( clscode ,currentTimeInMins );
}
```

■

Die eigentliche Berechnung der diversen Rollenausprägungen erfolgt durch Umsetzung der in Kapitel 10.3.1 definierten Rechenvorschriften. Die dafür benötigten Koeffizienten (a und b) sind – wie auch die Ergebnisvariablen – in entsprechenden Arrays (blockweise nach CLS-Codes angeordnet) hinterlegt, so dass die Berechnungen wie im Folgenden am Beispiel der kumulierten Sitzungsdaten illustriert umgesetzt werden können.

Beispiel 10.3: Umsetzung der Rollenanalyse: Klasse `VLernendenModell`

```
void VLernendenModell::berechneRollenSitzungsdatenKumuliert( int clscode )
{
  // Ermittle Indizes fuer kumulierte Sitzungsdaten und CLS-Code
  int [] codeIndizes = getIndizesSDKum( code );

  // Fuer jeden Index index im Feld codeIndizes tue folgendes:
  for( int index : codeIndizes )
  {
```

```
    // Berechne eine Rollenauspraegung
    rolleSDKum[index] = (clsZaehler[clsCode] - koeffASDKum[index])
                      / koeffBSDKum[index];
  }
}
```

■

Die Berechnungen der Rollenausprägungen auf Basis der beteiligungs- und zeitbezogenen Sitzungsdaten erfolgen in analoger Art und Weise.

10.3.3 Benutzertests mit Prototypen der Rollenanalyse

Wie die im Dezember 2005 zur Überprüfung der auf diese Weise berechneten Rollenausprägungen durchgeführt wurden, ist Bestandteil dieses Kapitels.

10.3.3.1 Beschreibung der Sitzungen und ihrer Teilnehmer

An diesen Benutzertests nahmen insgesamt 30 Studierende, verteilt auf zehn Gruppen teil. Von diesen zehn Gruppen bestanden acht Gruppen aus drei Teilnehmern und je eine Gruppe aus zwei beziehnugsweise vier Teilnehmern. Jeder Teilnehmer nahm im Rahmen dieser Benutzertests an genau einer Sitzung teil. Die genaue Verteilung der Teilnehmer auf die Sitzunge kann der Tabelle 10.11 entnommen werden.

Tab. 10.11: Benutzertests zur Evaluierung der Rollenanalyse

Sitzung	**Teilnehmer**	**Gruppe**	**Dauer**
S_{58}	T_{291}, T_{292}, T_{293}	G_{12}	94
S_{59}	T_{285}, T_{286}, T_{287}	G_{13}	102
S_{60}	T_{308}, T_{309}, T_{310} / T_{307}	G_{14}	112
S_{61}	T_{300}, T_{302} / T_{301}	G_{15}	105
S_{62}	T_{209}, T_{299}	G_{16}	81
S_{63}	T_{311}, T_{312} / T_{313}	G_{17}	118
S_{64}	T_{294}, T_{296} / T_{295}	G_{18}	105
S_{65}	T_{315}, T_{316} / T_{317}	G_{19}	76
S_{66}	T_{303}, T_{304}, T_{305}	G_{20}	103
S_{67}	T_{297}, T_{298} / T_{314}	G_{21}	101
10	24 / 6	10	$\phi_{ari} = 99,7$

24 der 30 Teilnehmer beantworteten im Anschluss an die Benutzertests den elektronischen Rollenfragebogen nach Spencer & Pruss (1995) aus (s. Kap.9.2.2.1) und lieferten damit Rollenprofile ab, die ebenfalls für die Evaluierung des Rollenanalyse-Prototypen verwendet wurden.

10.3.3.2 Aufgabe und Sitzungsverlauf

Alle Gruppen hatten dieselbe Aufgabe zu bearbeiten, die – wie schon in vorigen Benutzertests (s. Kap.10.1.1.1.2) – hinsichtlich Inhalt und Schwierigkeitsgrad an zuvor behandelte Lehrinhalte und den daraus folglich zu erwartenden Kenntnisstand der Teilnehmer angepasst war. Auch die Aufgabenverteilung war

Der Sitzungsablauf war analog zu den bereits in den Kapiteln 8.1.2.2 und 10.1.1.1.3 dargelegten Ausführungen organisiert, so dass auch hier für die eigentliche Aufgabenbearbeitung wieder 100 Minuten eingeplant wurden. Wie in Tabelle 10.11 zu sehen ist, betrug die durchschnittliche Sitzungsdauer 99,7 Minuten. Größere Abweichungen (von mehr als ± 10 % $= \pm 10$ Minuten) bei einzelnen Sitzungen sind durch meist erhebliche Probleme der Teilnehmer mit den Lehrinhalten (S_{60}, S_{62} und S_{65}) beziehungsweise mit den Bedienkonzepten (S_63) erklärbar. Diese Probleme führten in zwei Fällen (S_{60} und S_{63}) zu Verzögerungen bei der Aufgabenbearbeitung, in zwei anderen Fällen (S_{62} und S_{65}) blieben selbst geringe Erfolgserlebnisse weitestgehend aus, was zu sinkender Motivation verbunden mit einem frühzeitigen Sitzungsende führte.

10.4 Resultate der Evaluierung

Die aus den Sitzungen S_{58} bis S_{67} erzielten Ergebnisse in Form von Rollenausprägungen, die während der Sitzungen aus den von den Teilnehmern ausgelösten Aktionen beziehungsweise deren CLS-Codes in der prototypischen Analysekomponente berechnet wurden, sind den jeweiligen per Fragebogen ermittelten Rollenausprägungen der Teilnehmern gegenüberzustellen. Aus diesen Betrachtungen heraus ergeben sich weitere Diskussionspunkte, die die originären Ergebnisse um interessante Aspekte, auch im Hinblick auf mögliche Weiterentwicklungen, ergänzen.

10.4.1 Berechnete Rollenausprägungen

Anstatt sämtliche Resultate, die sich aus den Berechungen auf Grundlage der 88 aufgedeckten linearen Zusammenhänge (s. Tab.10.10) und deren Koeffizienten für die lineare Regression (s. Tab.D.2 bis D.7) ergeben, im Detail zu präsentieren, was sowohl den Rahmen sprengen würde als auch kaum sinnvoll erscheint, werden für jede Rolle jeweils die beiden Ergebnisse mit den für die Rolle höchsten Bestimmtheitsmaßen diskutiert. Die Vorstellung der Rollen und ihrer Ergebnisse erfolgt mit absteigender Relevanz der Rollen wie in Tabelle 9.5 festgelegt.

10.4.1.1 Resultate der Rolle *Informationsbeschaffer*

Wie bereits in Kapitel 10.2.1 ausgeführt wurde, sind die linearen Zusammenhaänge zwischen den Rollenausprägungen des Informationsbeschaffers (Rolle R_1) und den aus den betrachteten Sitzungen (S_1 - S_{27}) resultierenden Code-Kennzahlen nur sehr

schwach ausgeprägt, da sämtliche zugehörige Bestimmtheitsmaße deutlich unter 0,2 liegen. Von den 12 signifikanten Ergebnissen des Informationsbeschaffers erzielt die Betrachtung von CLS-Code $c_l = 24$ mit $B^b_{1,24} = 0,1843$ bei den bereinigten kumulierten Sitzungsdaten und mit $B^f_{1,24} = 0,1774$ bei den bereinigten zeitbezogenen Sitzungsdaten die beiden höchsten Bestimmtheitsmaße.

10.4.1.1.1 Bereinigte kumulierte Daten, Code 24 Die Ergebnisse wurden mit den zugehörigen Koeffizienten $a^b_{1,24} = 5,9494$ und $b^b_{1,24} = -0,1988$ (s. Tab.D.3) unter Einbeziehung der jeweiligen Code-Kennzahlen $C^b_{i,k,24}$ aus Sitzung S_i ($58 \leq i \leq 67$) von Teilnehmer T_k berechnet und in der folgenden Tabelle 10.12 zusammengefasst, aufsteigend sortiert nach per Fragebogen ermittelten Rollenausprägungen (Spalte $p_{k,1}$).

Tab. 10.12: Evaluierung der Rollenanalyse: Rolle *IB*, Code 24, Verfahren (b)

Lfd.Nr.	S_i	T_k	$C^b_{i,k,24}$	$p^b_{i,k,1,24}$	$p_{k,1}$	Δ_{abs}
1	S_{59}	T_{286}	2	19,87	9	10,87
2	S_{66}	T_{305}	1	24,91	11	13,91
3	S_{62}	T_{299}	3	14,84	12	2,84
4	S_{60}	T_{309}	1	24,91	14	10,91
5	S_{61}	T_{302}	1	24,91	15	9,91
6	S_{65}	T_{316}	3	14,84	16	1,16
7	S_{63}	T_{311}	1	24,91	17	7,91
8	S_{60}	T_{308}	11	-25,42	18	43,42
9	S_{64}	T_{296}	1	24,91	19	5,91
10	S_{59}	T_{287}	3	14,84	20	5,16
11	S_{66}	T_{303}	1	24,91	20	4,91
12	S_{60}	T_{310}	1	24,91	21	3,91
13	S_{59}	T_{285}	2	19,87	22	2,13
14	S_{67}	T_{297}	2	19,87	23	3,13

Deutlich sind die Unterschiede zwischen den per Fragebogen abgelieferten Rollenausprägungen (Spalte $p_{k,1}$) und den anhand der Sitzungsdaten berechneten Rollenausprägungen (Spalte $p^b_{i,k,1,24}$) erkennbar. Sie sind in der letzten Spalte (Δ_{abs}) aufgeführt und reichen von 1,16 Punkten (Zeile 6) bis hin zu 43,42 Punkten (Zeile 8) und betragen durchschnittlich 9,0057 Punkte.

Die Abbildung 10.5 veranschaulicht diese Differenzen. Entsprechend der Nummerierung in der Tabelle 10.12 repräsentieren die eckigen Kästchen die per Fragebogen ermittelten Ausprägungswerte ($p_{k,1}$) hinsichtlich der Informationsbeschaffer-Rolle, wohingegen die in der Analysekomponente errechneten Rollenausprägungen durch Kreise markiert sind.

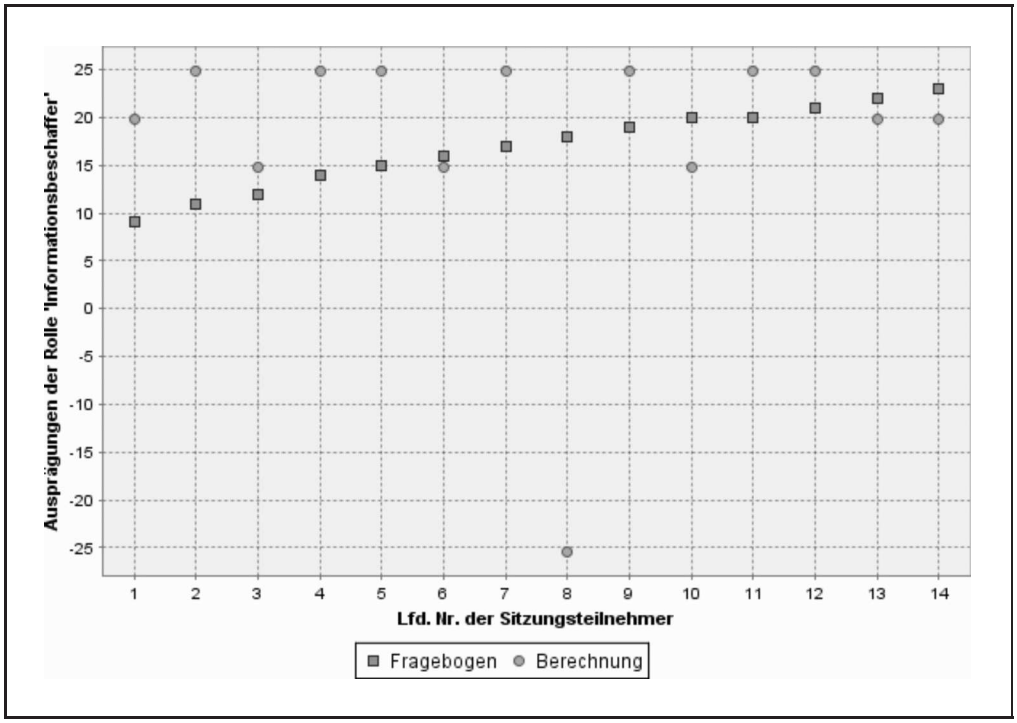

Abb. 10.5: Gegenüberstellung von Rollenausprägungen: Rolle *IB*, Code 24

Eine Korrelationsanalyse, die anhand der in der Abbildung 10.5 dargestellten Stichprobe durchgeführt wurde, bestätigt die Vermutung, dass zwischen den per Fragebogen erzielten Ausprägungswerten des Informationsbeschaffers und den anhand der hier betrachteten Verwendung von CLS-Code 24 kein statistisch nachweisbarer linearer Zusammenhang besteht: Ein Bestimmtheitsmaß von $B = 0,0033$ bei einem Stichprobenumfang von $N = 14$ belegt in diesem Fall, dass die Ergebnisse der Rollenanalyse anhand der durchgeführten Benutzertests nicht bestätigt werden können.

Für diesen speziellen Fall kann die in Kapitel 7.3.4.2 formulierte These nicht bestätigt werden. Bevor jedoch aus diesem (Teil-)Ergebnisse Schlüsse gezogen werden, werden die weiteren Resultate der Evaluierung diskutiert.

10.4.1.1.2 Bereinigte zeitbezogene Daten, Code 24 Ein ähnliches Bild ergibt die Überprüfung von Code 24 auf Basis der bereinigten zeitbezogenen Sitzungsdaten $C^f_{i,k,24}$ für die Rolle des Informationsbeschaffers. Die mit den Koeffizienten $a^f_{1,24} = 6,9$ und $b^f_{1,24} = -0,24$ (s. Tab.D.7) errechneten Rollenausprägungen $p^f_{i,k,1,24}$ sind den per Fragebogen erzielten Werten $p_{k,1}$ in der Tabelle 10.13 gegenübergestellt.

Auch hierbei sind Abweichungen zwischen Fragebogenergebnissen und per Analysekomponente berechneten Profilausprägungen ersichtlich. Aufgrund dieser Abweichungen kann auf eine weitere Betrachtung dieser Werte und weiterer für den Informationsbeschaffer berechneter Ausprägungswerte verzichtet werden.

Tab. 10.13: Evaluierung der Rollenanalyse: Rolle *IB*, Code 24, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,24}$	t_i	$C^f_{i,k,24}$	$p^f_{i,k,1,24}$	$p_{k,1}$	Δ_{abs}
1	S_{59}	T_{286}	2	102	1,9608	20,56	9	11,56
2	S_{66}	T_{305}	1	103	0,9709	24,68	11	13,68
3	S_{62}	T_{299}	3	81	3,7037	13,31	12	1,31
4	S_{60}	T_{309}	1	112	0,8929	25,01	14	11,01
5	S_{61}	T_{302}	1	105	0,9524	24,76	15	9,76
6	S_{65}	T_{316}	3	76	3,9474	12,29	16	3,71
7	S_{63}	T_{311}	1	118	0,8475	25,2	17	8,2
8	S_{60}	T_{308}	11	112	9,8214	-12,16	18	30,16
9	S_{64}	T_{296}	1	105	0,9524	24,76	19	5,76
10	S_{59}	T_{287}	3	102	2,9412	16,48	20	3,52
11	S_{66}	T_{303}	1	103	0,9709	24,68	20	4,68
12	S_{60}	T_{310}	1	112	0,8929	25,01	21	4,01
13	S_{59}	T_{285}	2	102	1,9608	20,56	22	1,44
14	S_{67}	T_{297}	2	101	1,9802	20,48	23	2,52

Auch ohne eine weitere Überprüfung sollte klar erkennbar sein, dass die Ergebnisse der Rollenanalyse anhand der durchgeführten Benutzertests für die Rolle des Informationsbeschaffers nicht bestätigt werden können. Da die Probanden beim Ausfüllen des Rollenfragebogens ja keinerlei Information hinsichtlich des Rollenmodells besaßen (s. Kap.9.2.2.2), kann eine Fehleinschätzung der Rollen durch die Teilnehmer ausgeschlossen werden. Vielmehr sind diese Abweichungen in den Ergebnissen dahingehend zu interpretieren, dass der vermutetete Zusammenhang zwischen Rollenausprägung der Rolle des Informationsbeschaffers und der Verwendung von Code 24 nicht existiert und folglich auch nicht nachgewiesen werden kann.

10.4.1.2 Resultate der Rolle *Planer*

Die Voruntersuchung des Planers hat insgesamt 8 verwertbare Ergebnisse geliefert (s. Tab.10.10), von denen hier die beiden mit den höchsten Bestimmtheitsmaßen anhand der Sitzungen S_{58} - S_{67} überprüft werden sollen.

10.4.1.2.1 Bereinigte kumulierte Daten, Code 39 Mit einem Bestimmtheitsmaß von $B^b_{2,39} = 0,2499$ (bei einem Stichprobenumfang von $N = 63$) lieferten die bereinigten kumulierten Sitzungsdaten $C^b_{i,k,39}$ das aussagekräftigste Ergebnis innerhalb der Untersuchungen zur Rolle des Planers. Die zugehörigen Koeffizienten sind $a^b_{2,39} = -1,3404$ und $b^b_{2,39} = 0,165$ (s. Tab.D.3). Die mit den Code-Kennzahlen der Sitzungen S_{58} - S_{67} berechneten Ergebnisse sind in der der folgenden Tabelle wiedergegeben.

Tab. 10.14: Evaluierung der Rollenanalyse: Rolle *PL*, Code 29, Variante (b)

Lfd.Nr.	S_i	T_k	$C^b_{i,k,39}$	$p^b_{i,k,2,39}$	$p_{k,2}$	Δ_{abs}
1	S_{58}	T_{291}	1	14,18	13	1,18
2	S_{64}	T_{296}	1	14,18	13	1,18
3	S_{65}	T_{315}	1	14,18	13	1,18
4	S_{58}	T_{293}	1	14,18	14	0,18
5	S_{59}	T_{286}	2	20,24	15	5,24
6	S_{60}	T_{308}	2	20,24	16	4,24
7	S_{61}	T_{300}	2	20,24	16	4,24
8	S_{62}	T_{299}	1	14,18	16	1,82
9	S_{66}	T_{305}	1	14,18	16	1,82
10	S_{S61}	T_{302}	3	26,3	17	9,3
11	S_{63}	T_{312}	2	20,24	17	3,24
12	S_{64}	T_{294}	3	26,3	17	9,3
13	S_{65}	T_{316}	1	14,18	17	2,82
14	S_{67}	T_{297}	3	26,3	17	9,3
15	S_{58}	T_{292}	1	14,18	18	3,82
16	S_{60}	T_{309}	1	14,18	19	4,82
17	S_{67}	T_{298}	1	14,18	21	6,82
18	S_{62}	T_{290}	2	20,24	22	1,76
19	S_{66}	T_{303}	1	14,18	22	7,82
20	S_{59}	T_{287}	1	14,18	23	8,82
21	S_{59}	T_{285}	2	20,24	24	3,76
22	S_{60}	T_{310}	1	14,18	24	9,82

Mit einer durchschnittlichen Differenz von 4,6582 Punkten fallen die Abweichungen zwar geringer aus als bei den zuvor betrachteten Ergebnissen des Informationsbeschaffers, sind aber dennoch deutlich größer als 0. Somit muss auch für dieses Teilergebnis festgestellt werden, dass die aus den Sitzungsdaten der Benutzertests berechneten Rollenausprägungen von den per Fragebogen ermittelten Profilwerten deutlich abweichen, so dass von einer Näherung auch in diesem Fall nicht die Rede sein kann.

10.4.1.2.2 Bereinigte zeitbezogene Daten, Code 39 Das zweitbeste Ergebnis innerhalb der Voruntersuchungen zur Planer-Rolle konnte mit den bereinigten zeitbezogenen Sitzungsdaten $C^f_{i,k,39}$ ebenfalls für Code 39 erreicht werden, jedoch lassen sich mit einem Bestimmtheitsmaß von $B^f_{2,39} = 0,1699$ (bei einem Stichprobenumfang von $N = 63$) weniger als 20 % der Stichprobe durch die zugehörige Regressionsgerade mit den Koeffizienten $a^f_{2,39} = -0,9582$ und $b^f_{2,39} = 0,1479$ erklären. Dementsprechend fallen auch die Ergebnisse der berechneten Rollenausprägungen aus, die in der Tabelle 10.15 zusammengefasst sind.

Tab. 10.15: Evaluierung der Rollenanalyse: Rolle *PL*, Code 39, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,39}$	t_i	$C^f_{i,k,39}$	$p^f_{i,k,2,39}$	$p_{k,2}$	Δ_{abs}
1	S_{58}	T_{291}	1	94	1,0638	13,67	13	0,67
2	S_{64}	T_{296}	1	105	0,9524	12,92	13	0,08
3	S_{65}	T_{315}	1	76	1,3158	15,37	13	2,37
4	S_{58}	T_{293}	1	94	1,0638	13,67	14	0,33
5	S_{59}	T_{286}	2	102	1,9608	19,74	15	4,74
6	S_{60}	T_{308}	2	112	1,7857	18,55	16	2,55
7	S_{61}	T_{300}	2	105	1,9048	19,36	16	3,36
8	S_{62}	T_{299}	1	81	1,2346	14,83	16	1,17
9	S_{66}	T_{305}	1	103	0,9709	13,04	16	2,96
10	S_{61}	T_{302}	3	105	2,8571	25,8	17	8,8
11	S_{63}	T_{312}	2	118	1,6949	17,94	17	0,94
12	S_{64}	T_{294}	3	105	2,8571	25,8	17	8,8
13	S_{65}	T_{316}	1	76	1,3158	15,37	17	1,63
14	S_{67}	T_{297}	3	101	2,9703	26,56	17	9,56
15	S_{58}	T_{292}	1	94	1,0638	13,67	18	4,33
16	S_{60}	T_{309}	1	112	0,8929	12,52	19	6,48
17	S_{67}	T_{298}	1	101	0,9901	13,17	21	7,83
18	S_{62}	T_{290}	2	81	2,4691	23,17	22	1,17
19	S_{66}	T_{303}	1	103	0,9709	13,04	22	8,96
20	S_{59}	T_{287}	1	102	0,9804	13,11	23	9,89
21	S_{59}	T_{285}	2	102	1,9608	19,74	24	4,26
22	S_{60}	T_{310}	1	112	0,8929	12,52	24	11,48

Zusammenfassend sind auch die für die Rolle des Planers berechneten Rollenausprägungen kaum verwendbar. Dies wird bestätigt durch die Ergebnisse einer Korrelationsanalyse, die zwischen $p_{k,2}$ und $p^b_{i,k,2,39}$ (bzw. $p_{k,2}$ und $p^f_{i,k,2,39}$) durchgeführt wurde: Bei einem Bestimmtheitsmaß von $B = 0,0002$ (bzw. $B = 0,0$) und einem Stichprobenumfang von jeweils $N = 22$ muss ein vermuteter linearer Zusammenhang zwischen per Fragebogen ermittelter und per Analysekomponente berechneter Rollenausprägung abgelehnt werden.

10.4.1.3 Resultate der Rolle *Berater*

Wie bei den zuvor überprüften Rollen finden sich auch beim Berater die beiden Ergebnisse mit den höchsten Bestimmtheitsmaßen in den bereinigten kumulierten Sitzungsdaten und den bereinigten zeitbezogenen Sitzungsdaten, jeweils bei Betrachtung des CLS-Codes 32.

10.4.1.3.1 Bereinigte kumulierte Daten, Code 32 Die eingangs durchgeführte Untersuchung der bereinigten kumulierten Sitzungsdaten $C^b_{i,k,32}$ resultierte

für CLS-Code 32 in einem Bestimmtheitsmaß $B^b_{3,32} = 0,4123$ bei einem Stichprobenumfang $N = 20$. Die Berechnung der Rollenausprägung mit den zugehörigen Koeffizienten $a^b_{3,32} = 8,0343$ und $b^b_{3,32} = -0,2926$ der Regressionsanalyse (s. Tabelle D.3) unter Verwendung der entsprechenden Sitzungsdaten aus den Sitzungen S_{58} bis S_{67} führt zu den in Tabelle 10.16 dargestellten Ergebnissen.

Tab. 10.16: Evaluierung der Rollenanalyse: Rolle *BR*, Code 32, Variante (b)

Lfd.Nr.	S_i	T_k	$C^b_{i,k,32}$	$p^b_{i,k,3,32}$	$p_{k,3}$	Δ_{abs}
1	S_{66}	T_{303}	2	20,62	18	2,62
2	S_{59}	T_{286}	2	20,62	19	1,62
3	S_{58}	T_{293}	1	24,04	20	4,04
4	S_{63}	T_{311}	1	24,04	24	0,04
5	S_{59}	T_{285}	1	24,04	25	0,96
6	S_{59}	T_{287}	1	24,04	25	0,96
7	S_{60}	T_{310}	1	24,04	25	0,96

In Bezug auf die betrachtete Stichprobe ($N = 7$) ergibt sich eine durchschnittliche Abweichung von 1,6 Punkten. Im Vergleich mit allen anderen bislang überprüften Analyseergebnissen stellt dies die geringste Abweichung zwischen $p_{k,n}$ (per Fragebogen ermittelte Rollenausprägungen) und $p^b_{i,k,n,l}$ (per Analysekomponente berechnete Rollenausprägungen) einer bestimmten Rolle R_n und einer zu einem CLS-Code c_l gehörenden Code-Kennzahl. Eine auf dieser Stichprobe durchgeführte Korrelationsanalyse bestätigt die vergleichsweise hohe Güte dieses Ergebnisses mit einem Bestimmtheitsmaß $B = 0,6752$.

10.4.1.3.2 Bereinigte zeitbezogene Daten, Code 32 Ein ähnlich gutes Ergebnis liefert auch die Überprüfung anhand der bereinigten kumulierten Sitzungsdaten $C^f_{i,k,32}$, die in der ersten Analysephase ein Bestimmtheitsmaß $B^f_{3,32} = 0,3469$ ergaben (Stichprobenumfang $N = 20$). Die in Tabelle 10.17 dargestellten Ergebnisse wurden unter Verwendung der Koeffizienten $a^f_{2,39} = 8,8222$ und $b^f_{2,39} = -0,3259$ (s. Tab.D.7) berechnet.

Tab. 10.17: Evaluierung der Rollenanalyse: Rolle *BR*, Code 32, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,32}$	t_i	$C^f_{i,k,32}$	$p^f_{i,k,3,32}$	$p_{k,3}$	Δ_{abs}
1	S_{66}	T_{303}	2	103	1,9417	21,11	18	3,11
2	S_{59}	T_{286}	2	102	1,9608	21,05	19	2,05
3	S_{58}	T_{293}	1	94	1,0638	23,80	20	3,80
4	S_{63}	T_{311}	1	118	0,8475	24,47	24	0,47
5	S_{59}	T_{285}	1	102	0,9804	24,06	25	0,94
6	S_{59}	T_{287}	1	102	0,9804	24,06	25	0,94
7	S_{60}	T_{310}	1	112	0,8929	24,33	25	0,67

Es ergibt sich hier eine durchschnittliche Abweichung zwischen $p_{k,3}$ und $p^f_{i,k,3,32}$ von 1,7114 Punkten, die nur unwesentlich größer ist als die beim vorigen Ergebnis berechnete mittlere Abweichung von 1,6 Punkten. Die Korrelationsanalyse, die auf Basis der in Tabelle 10.17 dargestellten Ergebnisse durchgeführt wurde, liefert mit einem Bestimmheitsmaß $B = 0,741$ ein besseres Resultat (Vergleichswert: $B = 0,6752$) bei identischem Stichprobenumfang ($N = 7$). Die folgende Abbildung stellt die Stichprobe der vorigen Tabelle graphisch dar.

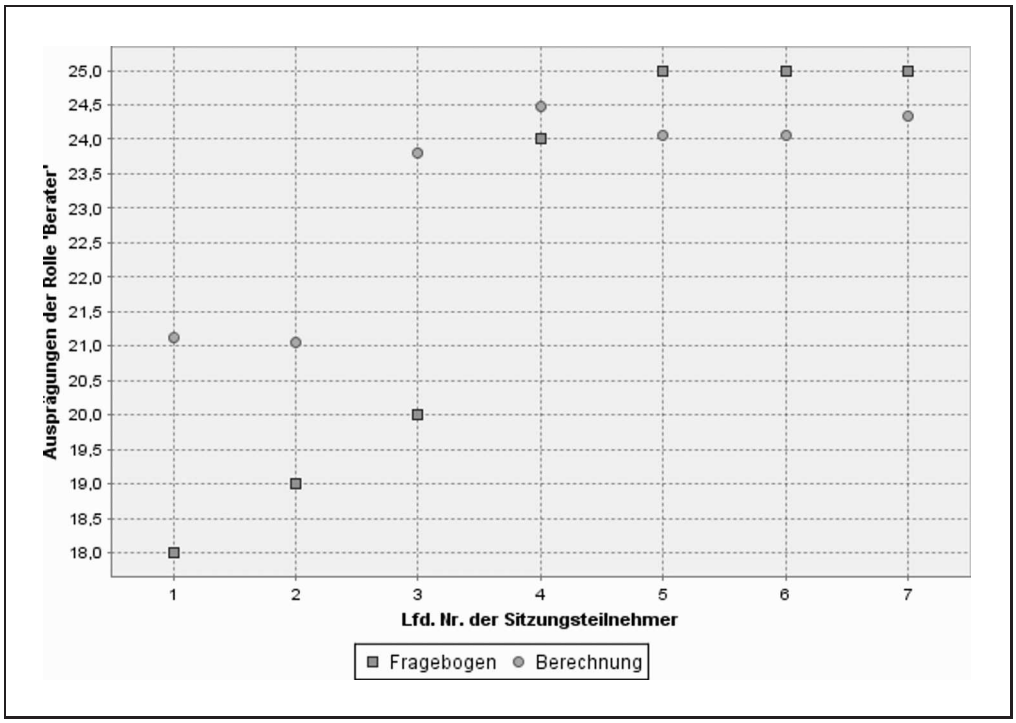

Abb. 10.6: Gegenüberstellung von Rollenausprägungen: Rolle *BR*, Code 32

Wie der Abbildung 10.6 zu entnehmen ist, werden auch hier die per Fragebogen ermittelten Ausprägungswerte der Berater-Rolle – unter Berücksichtigung der bereits genannten Differenzen zwischen $p_{k,3}$ und $p^f_{i,k,3,32}$ – nur grob durch die in der Analysekomponente berechneten Werte angenähert. Dennoch lässt sich aus dieser nur groben Annäherung bei entsprechender Sichtweise ein durchaus verwertbares Ergebnis ableiten: Bei Festlegung eines Grenzwertes, der zwischen niedrigen und hohen Rollenausprägungen unterscheidet, kann eine Gruppierung der Rollenausprägungen durchgeführt werden. Dieser Grenzwert ist im vorliegenden Beispiel zwischen 21,5 und 23,5 Punkten zu wählen, so dass diejenigen Teilnehmer, deren $p^f_{i,k,3,32}$ unter diesem Grenzwert liegt (T_{303} und T_{286}), in die Gruppe niedriger Berater-Ausprägung eingeordnet werden und Teilnehmer mit $p^f_{i,k,3,32}$ oberhalb des Grenzwerts (T_{311}, T_{285}, T_{287} und T_{310}) in die Gruppe hoher Berater-Ausprägung eingestuft werden. Der Teilnehmer T_{293} stellt eine Art Ausreisser dar: Aufgrund des berechneten $p^f_{58,293,3,32} = 23,8$ wäre dieser als Teilnehmer mit hoher Berater-Ausprägung zu klassifizieren, laut Fragebogenergebnis $p_{293,3} = 20$ ist er jedoch der Gruppe mit niedriger Berater-Ausprägung zuzuordnen. Zur Klärung dieses Fehlerfalls – und damit zur Festlegung geeigneter Maßnahmen zu dessen Bereinigung –

wären weitere Untersuchungen der entsprechenden Rollenprofile und Sitzungsdaten notwendig: So wäre die Frage nach dem Einfluss der Sitzungsdauer, die ja geringen Schwankungen unterliegt (s. Tab.10.11), eine mögliche Fragestellung in diesem Zusammenhang. Diese und weitere Untersuchungen sind jedoch nicht Gegenstand der vorliegenden Arbeit, sondern bei Bedarf in nachgelagerten Forschungsarbeiten durchzuführen.

10.4.1.4 Resultate der Rolle *Umsetzer*

Die Ergebnisse mit den beiden höchsten Bestimmtheitsmaßen sind bei Betrachtung der Umsetzer-Rolle unter den bereinigten zeitbezogenen Sitzungsdaten mit Code 41 ($B^f_{4,41} = 0,2063$) und Code 11 ($B^f_{4,11} = 0,1897$) zu finden.

10.4.1.4.1 Bereinigte zeitbezogene Daten, Code 41 Das höchste Bestimmtheitsmaß $B^f_{4,41} = 0,2063$(bei einem Stichprobenumfang von $N = 56$) wurde bei der Untersuchung der bereinigten kumulierten Sitzungsdaten $C^f_{i,k,41}$ erzielt. Die zugehörigen Koeffizienten sind $a^f_{4,41} = -25,7672$ und $b^f_{4,41} = 1,855$ (s. Tab.D.7). Die mit ihnen berechneten Ergebnisse aus den Evaluierungssitzungen S_{58} bis S_{67} sind der folgenden Tabelle 10.18 zu entnehmen.

Tab. 10.18: Evaluierung der Rollenanalyse: Rolle *UM*, Code 41, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,41}$	t_i	$C^f_{i,k,41}$	$p^f_{i,k,4,41}$	$p_{k,4}$	Δ_{abs}
1	S_{66}	T_{304}	20	103	19,4175	24,36	13	11,36
2	S_{64}	T_{296}	51	105	48,5714	40,07	14	26,07
3	S_{61}	T_{300}	7	105	6,6667	17,48	16	1,48
4	S_{62}	T_{299}	2	81	2,4691	15,22	16	0,78
5	S_{60}	T_{308}	17	112	15,1786	22,07	17	5,07
6	S_{66}	T_{303}	11	103	10,6796	19,65	17	2,65
7	S_{58}	T_{293}	4	94	4,2553	16,18	18	1,82
8	S_{60}	T_{310}	82	112	73,2143	53,36	18	35,36
9	S_{58}	T_{292}	2	94	2,1277	15,04	19	3,96
10	S_{63}	T_{312}	20	118	16,9492	23,03	19	4,03
11	S_{65}	T_{315}	1	76	1,3158	14,6	19	4,4
12	S_{62}	T_{290}	10	81	12,3457	20,55	20	0,55
13	S_{64}	T_{294}	33	105	31,4286	30,83	20	10,83
14	S_{65}	T_{316}	4	76	5,2632	16,73	21	4,27
15	S_{59}	T_{285}	5	102	4,902	16,53	22	5,47
16	S_{59}	T_{286}	16	102	15,6863	22,35	22	0,35
17	S_{59}	T_{287}	11	102	10,7843	19,7	22	2,3
18	S_{66}	T_{305}	17	103	16,5049	22,79	22	0,79
19	S_{61}	T_{302}	18	105	17,1429	23,13	23	0,13

Fortsetzung auf nächster Seite

Tab. 10.18: Evaluierung der Rollenanalyse: Rolle *UM*, Code 41, Variante (f) *Forts.*

Lfd.Nr.	S_i	T_k	$C^a_{i,k,41}$	t_i	$C^f_{i,k,41}$	$p^f_{i,k,4,41}$	$p_{k,4}$	Δ_{abs}
20	S_{63}	T_{311}	13	118	11,0169	19,83	23	3,17
21	S_{67}	T_{297}	1	101	0,9901	14,42	23	8,58
22	S_{67}	T_{298}	21	101	20,7921	25,1	23	2,1
23	S_{58}	T_{291}	10	94	10,6383	19,63	24	4,37

Die Berechnung des linearen Zusammenhangs zwischen den beiden Arten von Profilausprägungen (Spalten $p^f_{i,k,4,41}$ und $p_{k,4}$) bestätigt mit einem Bestimmtheitsmaß von $B = 0,0632$ die eher geringe Aussagekraft der Ergebnisse und damit verbunden auch der zugehörigen Rollenanalyse.

10.4.1.4.2 Bereinigte zeitbezogene Daten, Code 11 Auch das mit einem Bestimmtheitsmaß $B^f_{4,11} = 0,1807$ (Stichprobenumfang $N = 41$) zweithöchste Ergebnis, basierend auf den bereinigten kumulierten Sitzungsdaten $C^f_{i,k,11}$, liefert kein besseres Analyseergebnis. Die Berechnungen mit den Koeffizienten $a^f_{4,11} = -3,2443$ und $b^f_{4,1} = 0,2645$ (s. Tab.D.7) sind in der folgenden Tabelle zusammengefasst.

Tab. 10.19: Evaluierung der Rollenanalyse: Rolle *UM*, Code 11, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,11}$	t_i	$C^f_{i,k,11}$	$p^f_{i,k,4,11}$	$p_{k,4}$	Δ_{abs}
1	S_{64}	T_{296}	2	105	1,9048	19,47	14	5,47
2	S_{62}	T_{299}	2	81	2,4691	21,6	16	5,6
3	S_{66}	T_{303}	1	103	0,9709	15,94	17	1,06
4	S_{58}	T_{293}	1	94	1,0638	16,29	18	1,71
5	S_{60}	T_{310}	24	112	21,4286	93,27	18	75,27
6	S_{58}	T_{292}	1	94	1,0638	16,29	19	2,71
7	S_{63}	T_{312}	1	118	0,8475	15,47	19	3,53
8	S_{65}	T_{315}	6	76	7,8947	42,11	19	23,11
9	S_{60}	T_{309}	9	112	8,0357	42,64	20	22,64
10	S_{62}	T_{290}	1	81	1,2346	16,93	20	3,07
11	S_{64}	T_{294}	5	105	4,7619	30,27	20	10,27
12	S_{65}	T_{316}	2	76	2,6316	22,21	21	1,21
13	S_{59}	T_{285}	2	102	1,9608	19,68	22	2,32
14	S_{59}	T_{286}	1	102	0,9804	15,97	22	6,03
15	S_{59}	T_{287}	1	102	0,9804	15,97	22	6,03
16	S_{66}	T_{305}	1	103	0,9709	15,94	22	6,06
17	S_{61}	T_{302}	1	105	0,9524	15,87	23	7,13
18	S_{58}	T_{291}	2	94	2,1277	20,31	24	3,69

Weitere signifikante Ergebnisse ergaben sich für die soeben betrachteten CLS-Codes 41 und 11 auch bei Untersuchung der kumulierten Sitzungsdaten ($B^a_{4,11} = 0,1532$, $B^a_{4,41} = 0,1079$), der bereinigten kumulierten Sitzungsdaten ($B^b_{4,11} = 0,1223$, $B^b_{4,41} = 0,1580$), der bereinigten beteiligungsbezogenen Sitzungsdaten ($B^d_{4,41} = 0,1236$) sowie der zeitbezogenen Sitzungsdaten ($B^e_{4,11} = 0,1589$, $B^e_{4,41} = 0,1351$), aber jeweils mit zum Teil deutlich geringeren Bestimmtheitsmaßen, weswegen diese hier nicht weiter betrachtet werden.

10.4.1.5 Resultate der Rolle *Schlichter*

Aus den Ergebnissen der Voruntersuchung tritt bei Betrachtung des Schlichters der Code 47 hervor, der sowohl bei den bereinigten kumulierten Sitzungsdaten mit $B^b_{5,47} = 0,3863$ als auch bei den bereinigten zeitbezogenen Sitzungsdaten mit $B^f_{5,47} = 0,4324$ wesentlich höhere Bestimmtheitsmaße erzielte als die übrigen CLS-Codes: Das dritthöchste Bestimmtheitsmaß mit $B^a_{5,24} = 0,1136$ für CLS-Code 24 bei Betrachtung der kumulierten Sitzungsdaten weist bereits einen deutlichen Abstand zu den beiden erstgenannten Ergebnissen auf und kann nur noch wenig mehr als 11 % der Stichprobe durch die zugehörige Regressionsgerade erklären.

10.4.1.5.1 Bereinigte zeitbezogene Daten, Code 47 Die Überprüfung der Analyseergebnisse für die bereinigten zeitbezogenen Sitzungsdaten $C^f_{i,k,47}$ mit den Koeffizienten $a^f_{5,47} = 7,9029$ und $b^f_{5,47} = -0,2453$ (s. Tab.D.7) liefert die in der folgenden Tabelle 10.20 Rollenausprägungen $p^f_{i,k,5,47}$.

Tab. 10.20: Evaluierung der Rollenanalyse: Rolle *SL*, Code 47, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,47}$	t_i	$C^f_{i,k,47}$	$p^f_{i,k,5,47}$	$p_{k,5}$	Δ_{abs}
1	S_{62}	T_{299}	12	81	14,8148	-28,18	17	45,18
2	S_{66}	T_{303}	9	103	8,7379	-3,4	17	20,4
3	S_{66}	T_{305}	11	103	10,6796	-11,32	18	29,32
4	S_{59}	T_{286}	10	102	9,8039	-7,75	19	26,75
5	S_{59}	T_{287}	19	102	18,6275	-43,72	20	63,72
6	S_{61}	T_{300}	4	105	3,8095	16,69	20	3,31
7	S_{62}	T_{290}	6	81	7,4074	2,02	20	17,98
8	S_{61}	T_{302}	20	105	19,0476	-45,43	21	66,43
9	S_{63}	T_{312}	42	118	35,5932	-112,88	21	133,88
10	S_{64}	T_{296}	62	105	59,0476	-208,48	21	229,48
11	S_{66}	T_{304}	21	103	20,3883	-50,89	21	71,89
12	S_{58}	T_{291}	16	94	17,0213	-37,17	22	59,17
13	S_{58}	T_{292}	4	94	4,2553	14,87	23	8,13
14	S_{60}	T_{310}	93	112	83,0357	-306,27	23	329,27
15	S_{63}	T_{311}	22	118	18,6441	-43,78	23	66,78

Fortsetzung auf nächster Seite

Tab. 10.20: Evaluierung der Rollenanalyse: Rolle *SL*, Code 47, Variante (f) *Forts.*

Lfd.Nr.	S_i	T_k	$C^a_{i,k,47}$	t_i	$C^f_{i,k,47}$	$p^f_{i,k,5,47}$	$p_{k,5}$	Δ_{abs}
16	S_{65}	T_{316}	1	76	1,3158	26,85	23	3,85
17	S_{60}	T_{308}	20	112	17,8571	-40,58	24	64,58
18	S_{59}	T_{285}	14	102	13,7255	-23,73	25	48,73
19	S_{64}	T_{294}	31	105	29,5238	-88,13	25	113,13
20	S_{67}	T_{298}	23	101	22,7723	-60,61	25	85,61

Wie man der Tabelle 10.20 deutlich entnehmen kann, ergeben sich zum Teil sehr große Differenzen zwischen den beiden Rollenausprägungen $p^f_{i,k,5,47}$ und $p_{k,5}$: Die kleinste Abweichung innerhalb der Stichprobe beträgt 3,31 Punkte (Zeile 6), die größte Abweichung beträgt hingegen 329,27 Punkte (Zeile 14), es ergibt sich eine durchschnittliche Differenz von 74,3795 Punkten.

10.4.1.5.2 Bereinigte kumulierte Daten, Code 47 Die auf der Basis der bereinigten kumulierten Sitzungsdaten $C^b_{i,k,47}$ mit den Koeffizienten $a^b_{5,47} = 7,1942$ und $b^b_{5,47} = -0,2262$ (s. Tab.D.3) berechneten Rollenausprägungen samt zugehörige Fragebogenergebnisse sind in der Tabelle 10.21 notiert.

Tab. 10.21: Evaluierung der Rollenanalyse: Rolle *SL*, Code 47, Variante (b)

Lfd.Nr.	S_i	T_k	$C^b_{i,k,47}$	$p^b_{i,k,5,47}$	$p_{k,5}$	Δ_{abs}
1	S_{62}	T_{299}	12	-21,24	17	38,24
2	S_{66}	T_{303}	9	-7,98	17	24,98
3	S_{66}	T_{305}	11	-16,82	18	34,82
4	S_{59}	T_{286}	10	-12,4	19	31,4
5	S_{59}	T_{287}	19	-52,18	20	72,18
6	S_{61}	T_{300}	4	14,12	20	5,88
7	S_{62}	T_{290}	6	5,28	20	14,72
8	S_{61}	T_{302}	20	-56,6	21	77,6
9	S_{63}	T_{312}	42	-153,84	21	174,84
10	S_{64}	T_{296}	62	-242,25	21	263,25
11	S_{66}	T_{304}	21	-61,02	21	82,02
12	S_{58}	T_{291}	16	-38,92	22	60,92
13	S_{58}	T_{292}	4	14,12	23	8,88
14	S_{60}	T_{310}	93	-379,27	23	402,27
15	S_{63}	T_{311}	22	-65,44	23	88,44
16	S_{65}	T_{316}	1	27,38	23	4,38
17	S_{60}	T_{308}	20	-56,6	24	80,6
18	S_{59}	T_{285}	14	-30,08	25	55,08
19	S_{64}	T_{294}	31	-105,22	25	130,22
20	S_{67}	T_{298}	23	-69,86	25	94,86

Auch hierbei treten Differenzen auf, die sich zwischen 4,38 Punkten (Zeile 16) und 402,27 Punkten (Zeile 14) bewegen und durchschnittlich 87,279 Punkte betragen. Die ansatzweise Überprüfung der beiden soeben vorgestellten Ergebnisse für Code 47 auf lineare Zusammenhänge zwischen den jeweiligen Rollenausprägungen mittels Korrelationsanalyse liefert Bestimmtheitsmaße von $B = 0,0559$ (bereinigte kumulierte Sitzungsdaten) und $B = 0,0498$ (bereinigte zeitbezogene Sitzungsdaten). Eine direkte Verwendbarkeit des Codes 47 zur Berechnung von Ausprägungen der Schlichter-Rolle ist somit abzulehnen.

10.4.1.6 Resultate der Rolle *Problemlöser*

Die Betrachtungen zur Rolle des Problemlösers lieferte für Code 30 mit den Bestimmtheitsmaßen $B^b_{6,30} = 0,8428$ und $B^f_{6,30} = 0,8244$ herausragende Ergebnisse.

10.4.1.6.1 Bereinigte kumulierte Daten, Code 30 Die Überprüfung der in den Sitzungen S_{58} bis S_{67} aufgezeichneten Sitzungsdaten mit den Koeffizienten $a^b_{6,30} = -3,7888$ und $b^b_{6,30} = 0,2609$ (s. Tab.D.3) berechneten liefert Rollenausprägungen, die deutlich von den jeweiligen Fragebogenwerten differieren. Wie der folgenden Tabelle 10.22 entnommen werden kann, reichen die Abweichungen von 0,36 Punkten (Zeilen 8-10) bis zu 21,69 Punkten (Zeile 1) und betragen durchschnittlich 7,2173 Punkte.

Tab. 10.22: Evaluierung der Rollenanalyse: Rolle *PB*, Code 30, Variante (b)

Lfd.Nr.	S_i	T_k	$C^b_{i,k,30}$	$p^b_{i,k,6,30}$	$p_{k,6}$	Δ_{abs}
1	S_{58}	T_{293}	5	33,69	12	21,69
2	S_{66}	T_{305}	3	26,02	12	14,02
3	S_{66}	T_{304}	1	18,36	13	5,36
4	S_{58}	T_{292}	3	26,02	14	12,02
5	S_{64}	T_{296}	1	18,36	15	3,36
6	S_{63}	T_{311}	2	22,19	17	5,19
7	S_{65}	T_{316}	4	29,86	17	12,86
8	S_{59}	T_{285}	1	18,36	18	0,36
9	S_{64}	T_{294}	1	18,36	18	0,36
10	S_{65}	T_{315}	1	18,36	18	0,36
11	S_{67}	T_{298}	2	22,19	26	3,81

Eine Überprüfung auf mögliche lineare Zusammenhänge zwischen den diversen Ausprägungswerten (Spalten $p^b_{i,k,6,30}$und $p_{k,6}$) bestätigt den aus den Abweichungen absehbaren Trend: Mit $B = 0,1127$ und einem Stichprobenumfang $N = 11$ kann ein solcher Zusammenhang ausgeschlossen werden.

10.4.1.6.2 Bereinigte zeitbezogene Daten, Code 30 Ähnliche Ergebnisse ergibt die Überprüfung für Code 30 anhand der bereinigten zeitbezogenen Sitzungsdaten $C^f_{i,k,47}$ mit den Koeffizienten $a^f_{6,30} = -3,7807$ und $b^f_{6,30} = 0,2591$ (s. Tab.D.7).

Tab. 10.23: Evaluierung der Rollenanalyse: Rolle *PB*, Code 30, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,30}$	t_i	$C^f_{i,k,30}$	$p^f_{i,k,6,30}$	$p_{k,6}$	Δ_{abs}
1	S_{58}	T_{293}	5	94	5,3191	35,12	12	23,12
2	S_{66}	T_{305}	3	103	2,9126	25,83	12	13,83
3	S_{66}	T_{304}	1	103	0,9709	18,34	13	5,34
4	S_{58}	T_{292}	3	94	3,1915	26,91	14	12,91
5	S_{64}	T_{296}	1	105	0,9524	18,27	15	3,27
6	S_{63}	T_{311}	2	118	1,6949	21,13	17	4,13
7	S_{65}	T_{316}	4	76	5,2632	34,9	17	17,9
8	S_{59}	T_{285}	1	102	0,9804	18,37	18	0,37
9	S_{64}	T_{294}	1	105	0,9524	18,27	18	0,27
10	S_{65}	T_{315}	1	76	1,3158	19,67	18	1,67
11	S_{67}	T_{298}	2	101	1,9802	22,23	26	3,77

Bei ähnlich hohen Abweichungen wie im vorigen Kapitel liefert auch die Überprüfung der bereinigten zeitbezogenen Sitzungsdaten auf lineare Zusammenhänge mit einem Bestimmtheitsmaß von $B = 0,0842$ keine Bestätigung der in der Voruntersuchung erzielten und in der prototypischen Analysekomponente umgesetzten Ergebnisse. Trotz der vergleichsweise sehr hohen Bestimmtheitsmaße von $B^b_{6,30} = 0,8428$ und $B^f_{6,30} = 0,8244$, die in der Voruntersuchung bei Betrachtung der Verwendung von CLS-Code 30 durch den Problemlöser ermittelt wurden, liefert die Evaluierung durch die Sitzungen S_{58} bis S_{67} somit eher ernüchternde Ergebnisse.

10.4.1.7 Resultate der Rolle *Fragesteller*

Aus den 10 signifikanten Ergebnissen zur Untersuchung der Rolle des Fragestellers ragt der CLS-Code 28 mit 2 Ergebnissen heraus.

10.4.1.7.1 Bereinigte kumulierte Daten, Code 28 Die Untersuchung der Verwendung von Code 28 innerhalb der bereinigten kumulierten Sitzungsdaten $C^b_{i,k,28}$ ergab ein Bestimmtheitsmaß $B^b_{7,28} = 0,3735$ (Stichprobenumfang $N = 19$. Die Überprüfung durch Berechnung entsprechender Ausprägungswerte p auf Grundlage der in den Sitzungen S_{58} bis S_{67} erhobenen Daten mit den Koeffizienten $a^b_{7,28} = -39,4071$ und $b^b_{7,28} = 2,2279$ (s. Tab.D.3) führte zu keiner Bestätigung der

Ergebnisse: Die für die in der folgenden Tabelle 10.24 aufgeführten Ausprägungswerte (Spalten $p^b_{i,k,7,28}$ und $p_{k,7}$) weisen bei einem Bestimmtheitsmaß $B = 0,0003$ keinen statistisch signifikanten linearen Zusammenhang auf.

Tab. 10.24: Evaluierung der Rollenanalyse: Rolle *FR*, Code 28, Variante (b)

Lfd.Nr.	S_i	T_k	$C^b_{i,k,28}$	$p^b_{i,k,7,28}$	$p_{k,7}$	Δ_{abs}
1	S_{59}	T_{286}	2	18,59	15	3,59
2	S_{58}	T_{293}	2	18,59	16	2,59
3	S_{66}	T_{304}	1	18,14	16	2,14
4	S_{59}	T_{285}	3	19,03	17	2,03
5	S_{66}	T_{305}	1	18,14	18	0,14
6	S_{64}	T_{294}	2	18,59	19	0,41
7	S_{64}	T_{296}	2	18,59	19	0,41
8	S_{63}	T_{311}	6	20,38	21	0,62
9	S_{60}	T_{309}	1	18,14	22	3,86
10	S_{65}	T_{315}	1	18,14	22	3,86
11	S_{58}	T_{291}	1	18,14	23	4,86
12	S_{66}	T_{303}	1	18,14	23	4,86
13	S_{59}	T_{287}	1	18,14	24	5,86

10.4.1.7.2 Bereinigte zeitbezogene Daten, Code 28 Auch das mit einem Bestimmtheitsmaß von $B^f_{7,28} = 0,3678$ zweithöchste Ergebnis der Voruntersuchung kann in den zur Überprüfung durchgeführten Sitzungen S_{58} bis S_{67} nicht bestätigt werden. In der Tabelle 10.25 sind die Ergebnisse zusammengefasst, die mittels der Koeffizienten $a^f_{7,28} = -38,7096$ und $b^f_{7,28} = 2,1889$ (s. Tab.D.7) aus den entsprechenden Sitzungsdaten errechnet wurden.

Tab. 10.25: Evaluierung der Rollenanalyse: Rolle *FR*, Code 28, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,28}$	t_i	$C^f_{i,k,28}$	$p^f_{i,k,7,28}$	$p_{k,7}$	Δ_{abs}
1	S_{59}	T_{286}	2	102	1,9608	18,58	15	3,58
2	S_{58}	T_{293}	2	94	2,1277	18,66	16	2,66
3	S_{66}	T_{304}	1	103	0,9709	18,13	16	2,13
4	S_{59}	T_{285}	3	102	2,9412	19,03	17	2,03
5	S_{66}	T_{305}	1	103	0,9709	18,13	18	0,13
6	S_{64}	T_{294}	2	105	1,9048	18,55	19	0,45
7	S_{64}	T_{296}	2	105	1,9048	18,55	19	0,45
8	S_{63}	T_{311}	6	118	5,0847	20,01	21	0,99
9	S_{60}	T_{309}	1	112	0,8929	18,09	22	3,91
10	S_{65}	T_{315}	1	76	1,3158	18,29	22	3,71

Fortsetzung auf nächster Seite

Tab. 10.25: Evaluierung der Rollenanalyse: Rolle *FR*, Code 28, Variante (f) *Forts.*

Lfd.Nr.	S_i	T_k	$C^a_{i,k,28}$	t_i	$C^f_{i,k,28}$	$p^f_{i,k,7,28}$	$p_{k,7}$	Δ_{abs}
11	S_{58}	T_{291}	1	94	1,0638	18,17	23	4,83
12	S_{66}	T_{303}	1	103	0,9709	18,13	23	4,87
13	S_{59}	T_{287}	1	102	0,9804	18,13	24	5,87

Die auf den per Fragebogen ermittelten Ausprägungswerten (Spalte $p_{k,7}$) und den per Analysekomponente errechneten Ausprägungswerten (Spalte $p^f_{i,k,7,28}$) angewandte Korrelationsanalyse liefert ein Bestimmtheitsmaß $B = 0,0373$ (bei einem Stichprobenumfang von $N = 13$), so dass ein linearer Zusammenhang zwischen den beiden Gruppen von Ausprägungswerten nicht angenommen werden kann.

10.4.1.8 Resultate der Rolle *Moderator*

Von den 6 statistisch nachweisbaren Ergebnissen der Untersuchungen der Moderator-Rollen erzielten CLS-Code 12 in den bereinigten zeitbezogenen Sitzungsdaten und CLS-Code 39 in den bereinigten kumulierten Sitzungsdaten mit Bestimmtheitsmaßen von $B^f_{8,12} = 0,3214$ und $B^b_{8,39} = 0,2039$ die beiden höchstwertigen Ergebnisse.

10.4.1.8.1 Bereinigte zeitbezogene Daten, Code 12 Die Überprüfung von CLS-Code 12 anhand der in den Sitzungen S_{58} bis S_{67} erfassten (bereinigten zeitbezogenen) Sitzungsdaten liefert die in der Tabelle 10.26 zusammengefassten Ausprägungswerte für die Moderator-Rolle.

Tab. 10.26: Evaluierung der Rollenanalyse: Rolle *MD*, Code 12, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,12}$	t_i	$C^f_{i,k,12}$	$p^f_{i,k,8,12}$	$p_{k,8}$	Δ_{abs}
1	S_{58}	T_{293}	1	94	1,0638	21,62	18	3,62
2	S_{60}	T_{309}	2	112	1,7857	26,26	18	8,26
3	S_{61}	T_{300}	1	105	0,9524	20,9	19	1,9
4	S_{62}	T_{290}	1	81	1,2346	22,72	19	3,72
5	S_{61}	T_{302}	1	105	0,9524	20,9	20	0,9
6	S_{66}	T_{305}	6	103	5,8252	52,2	21	31,2
7	S_{63}	T_{311}	3	118	2,5424	31,12	24	7,12
8	S_{67}	T_{298}	2	101	1,9802	27,51	25	2,51

Ein linearer Zusammenhang zwischen den jeweiligen Rollenausprägungen $p^f_{i,k,8,12}$ und $p_{k,8}$ eines Teilnehmers sind weder erkennbar noch statistisch nachweisbar: Eine zur Überprüfung eines linearen Zusammenhangs durchgeführte Korrelations-

analyse liefert für die achtelementige Stichprobe lediglich ein Bestimmtheitsmaß $B = 0,1041$.

10.4.1.8.2 Bereinigte kumulierte Daten, Code 39 Bei der Überprüfung der Verwendung von CLS-Code 39 durch den Moderator mittels der bereinigten kumulierten Sitzungsdaten ergeben sich die in folgender Tabelle 10.27 dargestellten Ergebnisse. Berechnungsgrundlage waren die Koeffizienten $a^b_{8,39} = -2,4319$ und $b^b_{8,39} = 0,1854$ (s. Tab.D.3).

Tab. 10.27: Evaluierung der Rollenanalyse: Rolle *MD*, Code 39, Variante (b)

Lfd.Nr.	S_i	T_k	$C^b_{i,k,39}$	$p^b_{i,k,8,39}$	$p_{k,8}$	Δ_{abs}
1	S_{66}	T_{303}	1	18,51	17	1,51
2	S_{58}	T_{293}	1	18,51	18	0,51
3	S_{60}	T_{309}	1	18,51	18	0,51
4	S_{59}	T_{285}	2	23,9	19	4,9
5	S_{61}	T_{300}	2	23,9	19	4,9
6	S_{62}	T_{290}	2	23,9	19	4,9
7	S_{63}	T_{312}	2	23,9	19	4,9
8	S_{60}	T_{310}	1	18,51	20	1,49
9	S_{61}	T_{302}	3	29,3	20	9,3
10	S_{58}	T_{292}	1	18,51	21	2,49
11	S_{60}	T_{308}	2	23,9	21	2,9
12	S_{65}	T_{316}	1	18,51	21	2,49
13	S_{66}	T_{305}	1	18,51	21	2,49
14	S_{59}	T_{287}	1	18,51	22	3,49
15	S_{64}	T_{296}	1	18,51	22	3,49
16	S_{65}	T_{315}	1	18,51	22	3,49
17	S_{67}	T_{297}	3	29,3	22	7,3
18	S_{59}	T_{286}	2	23,9	23	0,9
19	S_{62}	T_{299}	1	18,51	23	4,49
20	S_{64}	T_{294}	3	29,3	25	4,3
21	S_{67}	T_{298}	1	18,51	25	6,49
22	S_{58}	T_{291}	1	18,51	27	8,49

Wie den Ergebnissen der vorigen Tabelle zu entnehmen ist, sind diese mit großen Abweichungen behaftet, die von 0,26 Punkten (Zeile 2) bis zu 8,74 Punkten (Zeile 22) reichen und dabei entsprechende Streuung aufweisen. Demzufolge liefert auch eine zur Überprüfung dieser Ergebnisse durchgeführte Korrelationsanalyse mit $B = 0,0005$ einen nur sehr kleinen Wert, so dass ein vermuteter linearer Zusammenhang zwischen den Ausprägungswerten $p^b_{i,k,8,39}$ und $p_{k,8}$ auf jeden Fall abzulehnen ist.

10.4.1.9 Resultate der Rolle *Archivar*

Das einzige signifikante Ergebnis bei Betrachtung des Archivars wurde bei der Verwendung von CLS-Code 29 innerhalb der bereinigten beteiligungsbezogenen Sitzungsdaten $C^d_{i,k,29}$ ermittelt: Mit einem Bestimmtheitsmaß $B^d_{9,29} = 0,2571$ lassen sich gut 25 % der Stichproben durch die Regressionsgerade mit den Koeffizienten $a^d_{9,29} = -1,0237$ und $b^d_{9,29} = 0,0762$ (s. Tab.D.5) erklären. Bei der Überprüfung dieses Ergebnisses anhand der entsprechenden Sitzungsdaten aus den Sitzungen S_{58} bis S_{67} wurden Rollenausprägungen $p^d_{i,k,9,29}$ berechnet und den Fragebogenergebnissen $p_{k,9}$ der jeweiligen Teilnehmer gegenübergestellt (s. Tab.10.28).

Tab. 10.28: Evaluierung der Rollenanalyse: Rolle AR, Code 29, Variante (d)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,29}$	$A_{i,k}$	$C^d_{i,k,29}$	$p^d_{i,k,9,29}$	$p_{k,9}$	Δ_{abs}
1	S_{59}	T_{286}	2	1886	0,106	14,83	15	0,17
2	S_{60}	T_{308}	3	2169	0,1383	15,26	15	0,26
3	S_{60}	T_{309}	2	1602	0,1248	15,08	15	0,08
4	S_{66}	T_{303}	2	1686	0,1186	15	15	0
5	S_{58}	T_{291}	1	1247	0,0802	14,49	16	1,51
6	S_{60}	T_{310}	1	2498	0,04	13,97	16	2,03
7	S_{62}	T_{290}	3	2134	0,1406	15,29	16	0,71
8	S_{64}	T_{294}	1	3072	0,0326	13,87	16	2,13
9	S_{67}	T_{297}	1	478	0,2092	16,19	16	0,19
10	S_{58}	T_{292}	4	534	0,7491	23,28	17	6,28
11	S_{62}	T_{299}	3	350	0,8571	24,69	18	6,69
12	S_{65}	T_{315}	1	1066	0,0938	14,67	18	3,33
13	S_{66}	T_{304}	1	2333	0,0429	14	18	4
14	S_{66}	T_{305}	4	1368	0,2924	17,28	18	0,72
15	S_{63}	T_{312}	1	1726	0,0579	14,2	19	4,8
16	S_{65}	T_{316}	1	709	0,141	15,29	19	3,71
17	S_{67}	T_{298}	3	1570	0,1911	15,95	21	5,05

Auch diese Ergebnisse der Anlaysekomponente bestätigen die der Korrelationsanalyse nicht: Abweichungen zwischen 0,08 und 6,69 Punkten widersprechen einer brauchbaren Näherung der berechneten Rollenausprägungen $p^d_{i,k,9,29}$ an die per Fragebogen ermittelten Ausprägungswerte der Archivar-Rolle. Bestätigt wird dies durch das Ergebnis der auf diesen Werten durchgeführten Korrelationsanalyse: Bei einem Bestimmtheitsmaß von $B = 0,0312$ ist ein linearer Zusammenhang statistisch nicht belegbar.

10.4.1.10 Resultate der Rolle *Vertrauensperson*

Letztlich konnten auch bei den Untersuchungen zur Vertrauensperson zwei Ergebnisse erzielt werden, die in der prototypischen Analysekomponente umgesetzt und

anhand weiterer Benutzertests überprüft wurden.

10.4.1.10.1 Zeitbezogene Daten, Code 4 Die Analyse der zeitbezogenen Sitzungsdaten $C^e_{i,k,4}$ deckte lineare Zusammenhänge zwischen der Verwendung von CLS-Code 4 und den Ausprägungen der Rolle der Vertauensperson auf, deren Güte durch das Bestimmtheitsmaß $B^e_{10,4} = 0,1355$ (bei einem Stichprobenumfang von $N = 70$) beschrieben wird. Die Überprüfung anhand der in den Sitzungen S_{58} bis S_{67} gewonnenen Sitzungsdaten durch Berechnung der Rollenausprägungen $p^e_{i,k,10,4}$ mit den Koeffizienten $a^e_{10,4} = 10,8729$ und $b^e_{10,4} = -0,4523$ (s. Tab.D.6) lieferte die in Tabelle 10.29 aufgeführten Resultate.

Tab. 10.29: Evaluierung der Rollenanalyse: Rolle *VP*, Code 4, Variante (e)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,4}$	t_i	$C^e_{i,k,4}$	$p^e_{i,k,10,4}$	$p_{k,10}$	Δ_{abs}
1	S_{63}	T_{312}	2	118	1,6949	20,29	15	5,29
2	S_{66}	T_{303}	1	103	0,9709	21,89	15	6,89
3	S_{66}	T_{304}	1	103	0,9709	21,89	15	6,89
4	S_{66}	T_{305}	1	103	0,9709	21,89	15	6,89
5	S_{61}	T_{300}	4	105	3,8095	15,61	18	2,39
6	S_{62}	T_{290}	2	81	2,4691	18,58	18	0,58
7	S_{64}	T_{296}	2	105	1,9048	19,83	18	1,83
8	S_{58}	T_{292}	3	94	3,1915	16,98	19	2,02
9	S_{59}	T_{285}	10	102	9,8039	2,36	19	16,64
10	S_{65}	T_{316}	1	76	1,3158	21,13	19	2,13
11	S_{58}	T_{291}	2	94	2,1277	19,33	20	0,67
12	S_{59}	T_{286}	1	102	0,9804	21,87	20	1,87
13	S_{65}	T_{315}	3	76	3,9474	15,31	20	4,69
14	S_{67}	T_{297}	2	101	1,9802	19,66	20	0,34
15	S_{60}	T_{308}	2	112	1,7857	20,09	21	0,91
16	S_{60}	T_{309}	3	112	2,6786	18,12	21	2,88
17	S_{58}	T_{293}	2	94	2,1277	19,33	22	2,67
18	S_{59}	T_{287}	10	102	9,8039	2,36	22	19,64
19	S_{63}	T_{311}	1	118	0,8475	22,16	23	0,84
20	S_{64}	T_{294}	2	105	1,9048	19,83	23	3,17
21	S_{67}	T_{298}	2	101	1,9802	19,66	23	3,34
22	S_{60}	T_{310}	4	112	3,5714	16,14	24	7,86
23	S_{61}	T_{302}	7	105	6,6667	9,3	24	14,7
24	S_{62}	T_{299}	6	81	7,4074	7,66	24	16,34

Anhand der Differenzen, die sich zwischen 0,34 und 19,63 Punkten bewegen, ist schon absehbar, dass auch dieses Ergebnis keine brauchbare Näherung von $p^e_{i,k,10,4}$ an die Rollenausprägungen $p_{k,10}$ darstellt. Der einfache Test mit Berechnung des Bestimmtheitsmaßes bestätigt dies: Bei einem Bestimmtheitsmaß von $B = 0,1343$

und einem Stichprobenumfang von $N = 24$ ist kein linearer Zusammenhang zwischen den Rollenausprägungen $p^e_{i,k,10,4}$ und $p_{k,10}$ nachweisbar.

10.4.1.10.2 Kumulierte Daten, Code 4 Von allen hier überprüften Ergebnissen erreichte die Analyse der kumulierten Sitzungsdaten $C^a_{i,k,4}$ hinsichtlich der Verwendung von CLS-Code 4 durch die Rolle der Vertrauensperson mit einem Bestimmtheitsmaß $B^a_{10,4} = 0,1176$ (Stichprobenumfang $N = 70$) das geringwertigste Ergebnis. Die Überprüfung anhand der Sitzungen S_{58} bis S_{67} mit den Koeffizienten $a^a_{10,4} = 10,6367$ und $b^a_{10,4} = -0,4392$ (s. Tab.D.2) lieferte die in der folgenden Tabelle 10.30 aufgeführten berechneten Rollenausprägungen $p^a_{i,k,10,4}$.

Tab. 10.30: Evaluierung der Rollenanalyse: Rolle *VP*, Code 4, Variante (a)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,4}$	$p^a_{i,k,10,4}$	$p_{k,10}$	Δ_{abs}
1	S_{63}	T_{312}	2	19,66	15	4,66
2	S_{66}	T_{303}	1	21,94	15	6,94
3	S_{66}	T_{304}	1	21,94	15	6,94
4	S_{66}	T_{305}	1	21,94	15	6,94
5	S_{61}	T_{300}	4	15,11	18	2,89
6	S_{62}	T_{290}	2	19,66	18	1,66
7	S_{64}	T_{296}	2	19,66	18	1,66
8	S_{58}	T_{292}	3	17,39	19	1,61
9	S_{59}	T_{285}	10	1,45	19	17,55
10	S_{65}	T_{316}	1	21,94	19	2,94
11	S_{58}	T_{291}	2	19,66	20	0,34
12	S_{59}	T_{286}	1	21,94	20	1,94
13	S_{65}	T_{315}	3	17,39	20	2,61
14	S_{67}	T_{297}	2	19,66	20	0,34
15	S_{60}	T_{308}	2	19,66	21	1,34
16	S_{60}	T_{309}	3	17,39	21	3,61
17	S_{58}	T_{293}	2	19,66	22	2,34
18	S_{59}	T_{287}	10	1,45	22	20,55
19	S_{63}	T_{311}	1	21,94	23	1,06
20	S_{64}	T_{294}	2	19,66	23	3,34
21	S_{67}	T_{298}	2	19,66	23	3,34
22	S_{60}	T_{310}	4	15,11	24	8,89
23	S_{61}	T_{302}	7	8,28	24	15,72
24	S_{62}	T_{299}	6	10,56	24	13,44

Auch in diesem Fall werden die ursprünglich ermittelten Resultate leider nicht bestätigt: Die zum Teil großen Differenzen zwischen den Rollenausprägungen $p^a_{i,k,10,4}$ und $p_{k,10}$ lassen keinen linearen Zusammenhang zwischen diesen erkennen.

Das Bestimmtheitsmaß $B = 0,1265$ als Ergebnis einer einfachen Korrelationsanalyse bestätigt dies.

10.4.2 Ergänzende Betrachtungen

10.4.2.1 Lineare Zusammenhänge der Sitzungen S_{58}-S_{67}

Nachdem die Ergebnisse aus Kapitel 10.2 durch die aus den Sitzungen S_{58} bis S_{67} resultierenden Code-Kennzahlen nur in Ansätzen bestätigt werden konnten, wurde die in Kapitel 10.1 erläuterte Vorgehensweise auf die Sitzungen S_{58} bis S_{67} angewendet: Mit den protokollierten Sitzungsdaten und den Rollenprofilen der Teilnehmer wurden nach Abschluss der Sitzungen Korrelationsanalysen wie beschrieben durchgeführt, um ebenfalls statistisch nachweisbare lineare Zusammenhänge zwischen Rollenausprägungen und Code-Kennzahlen aufzudecken.

Das Ziel dieser Überprüfung bestand darin, diese neuen Ergebnisse den ursprünglichen Resultaten (s. Tab.10.10) gegenüberzustellen, zu diskutieren und somit weitere Erkenntnisse im Hinblick auf die Klärung der Frage nach dem Zusammenhang zwischen Code-Kennzahlen und Rollenausprägungen zu erlangen. Die signifikanten Ergebnisse der Korrelationsanalyse, basierend auf den Code-Kennzahlen der Sitzungen S_{58} bis S_{67} sind in der folgenden Tabelle 10.31 zusammengefasst.

Tab. 10.31: Signifikante Ergebnisse der Rollenanalyse (Sitzung S_{58}-S_{67})

Rolle	$C^a_{i,k}$	$C^b_{i,k}$	$C^c_{i,k}$	$C^d_{i,k}$	$C^e_{i,k}$	$C^f_{i,k}$
IB						
PL						5 : 0,5582
BR		33 : 0,8575		5 : 0,4634		
UM	15 : 0,2758				15 : 0,2757	
SL	26 : 0,2596					
PB				5 : 0,4745 13 : 0,5066		
FR		39 : 0,2842				39 : 0,2812
MD	5 : 0,3919				5 : 0,4004	
AR	4 : 0,3170	0 : 0,8250 4 : 0,3170		0 : 0,8755	4 : 0,3272	0 : 0,8337 4 : 0,3272
VP	10 : 0,2871 13 : 0,2810		45 : 0,2652		10 : 0,2861	

Zunächst fällt auf, dass im Vergleich zu den Sitzungen S_1 bis S_{27} wesentlich weniger signifikante lineare Zusammenhänge existieren: Den in den Sitzungen S_1 bis S_{27} aufgedeckten Ergebnissen stehen 23 Ergebnisse gegenüber, die sich aus der Betrachtung der Sitzungen S_{58} bis S_{67} ergeben. Von diesen 23 Resultaten fallen alleine 7 auf die Rolle des Archivars (S_1 - S_{27}: 1 Ergebnis), davon wiederum erzielen

3 Bestimmtheitsmaße zwischen 0, 82 und 0, 88, der Rest bewegt sich immerhin um die 0, 32. Ganz anders die Rolle des Informationsbeschaffers: Konnten in den Sitzungen S_1 bis S_{27} immerhin 11 Zusammenhänge nachgewiesen werden, so weisen die zur Überprüfung dieser Ergebnisse abgehaltenen Sitzungen kein verwertbares Resultat für den Informationsbeschaffer.

Ein ähnlich Bild ergibt auch die Untersuchungder anderen Rollen: Für Berater, Umnsetzer, Problemlöser und Fragesteller konnten bei der erneuten Durchführung der Rollenanalyse anhand der aus den Sitzungen S_{58} bis S_{67} resultierenden Code-Kennzahlen stets nur 2 signifikante Ergebnisse vorweisen statt zuvor 11 (BR), 18 (UM), 14 (PB) oder 10 (FR).

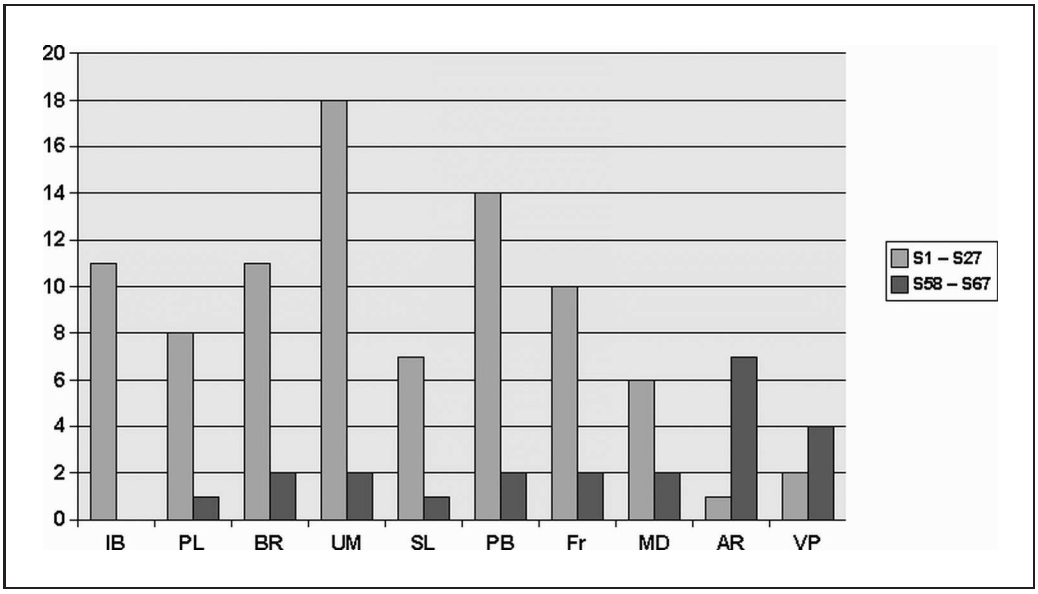

Abb. 10.7: Anzahl signifikanter Ergebnisse: S_1-S_{27} vs. S_{58}-S_{67}

Neben dieser rein quantitativen Betrachtung der Ergebnisse aus beiden Untersuchungen sind auch weitere Unterschiede feststellbar und nennenswert: Inhaltlich differieren die Tabellen 10.10 und 10.31 gänzlich voneinander. Dies äußert sich in der Tatsache, dass keiner der CLS-Codes bei Betrachtung einer bestimmten Rolle unter Verwendung der 6 verschiedenen Sitzungsdaten in der ursprünglichen Untersuchung (s. Tab.10.10) ein signifikantes Ergebnis liefert, das sich auch in der Überprüfung (s. Tab.10.31) wiederfindet - und umgekehrt. Darüber hinaus sind die Ergebnisse, die aus der Betrachtung der Sitzungen S_{58} bis S_{67} resultieren, mit tendentiell höheren Bestimmtheitsmaßen behaftet.

Das Beispiel des Archivars bei Verwendung von CLS-Code 0 unter Betrachtung der bereinigten beteiligungsbezogenen Sitzungsdaten – wie in Abbildung 10.8 dargestellt – veranschaulicht, wie eine brauchbare Näherung an Rollenausprägungen aussehen kann.

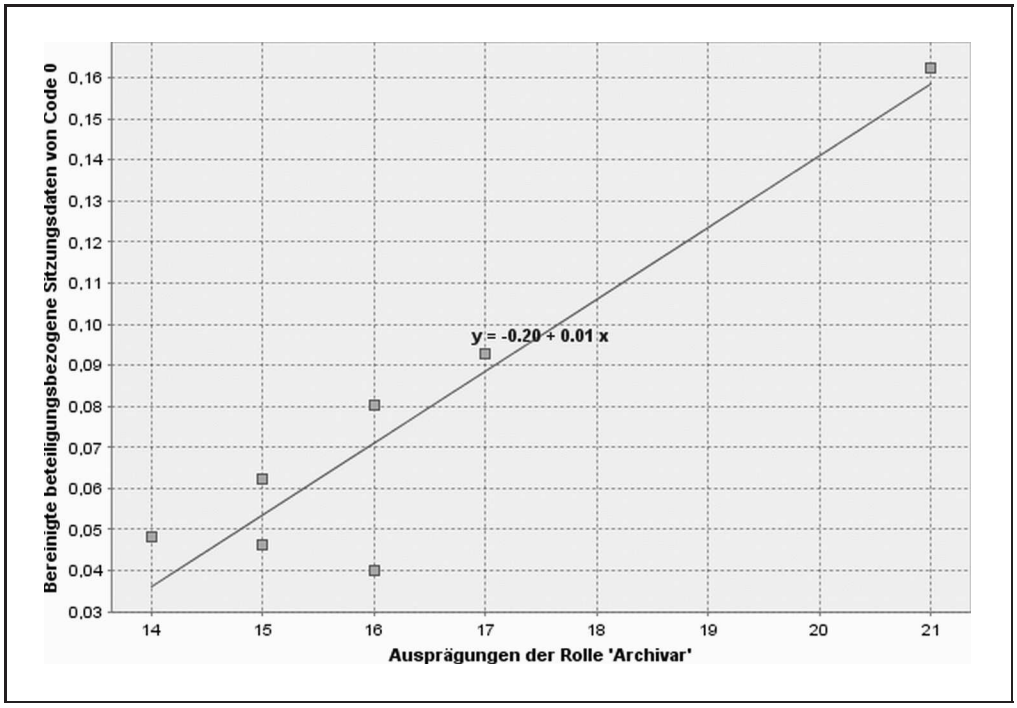

Abb. 10.8: Rollenanalyse (S_{58}-S_{67}): Rolle AR, Code 0, Variante (d)

Gründe für diese günstigeren Ergebnisse sind sicherlich auch in den Stichprobenumfängen zu suchen, dies jedoch ist bei Bedarf in Anschlussarbeiten näher zu analysieren.

10.4.2.2 Unerwartete Zusammenhänge

Nachdem bislang ausschließlich die signifikanten Ergebnisse betrachtet und diskutiert wurden, folgt – sowohl als Ergänzung wie auch als Abschluss – eine Fokussierung auf diejenigen Ergebnisse, die statistisch nicht begründbar sind: Analog zu den im Kapitel 10.3 durchgeführten Berechnungen von Rollenausprägungen $p^x_{i,k,n,l}$[7] anhand der zugehörigen Sitzungsdaten (respektive der entsprechenden Code-Kennzahlen $C^x_{i,k}$) wurden hierbei auch Rollenausprägungen $p^x_{i,k,n,l}$ mittels solcher Koeffizienten $a^x_{n,l}$ und $b^x_{n,l}$ berechnet, die zu einem nicht-signifikanten Analyseergebnis aus der Betrachtung von CLS-Code c_l und Rolle R_n resultieren. Mit den berechneten Rollenausprägungen $p^x_{i,k,n,l}$ und den zugehörigen per Fragebogen ermittelten Rollenausprägungen wird anschließend eine Korrelationsanalyse durchgeführt, anhand derer auf mögliche lineare Zusammenhänge zwischen den beiden Ausprägungsarten geschlossen werden kann. Als Indikator für die Güte dieses Zusammenhangs dient wiederum das Bestimmtheitsmaß B. Die Ergebnisse sind in der nachfolgenden Tabelle 10.32 zusammengefasst, absteigend sortiert nach dem Bestimmtheitsmaß B[8]. Die in kursiver Schrift notierten Einträge (Zeilen 2, 5, 8, 9, 34 und 41) beinhalten signifikante Ergebnisse, die auch in der Tabelle 10.31 wiederzufinden sind.

[7] Der Index x repräsentiert eine der 6 Varianten $a, b, \ldots, f$.

[8] Auf Resultate mit einem Bestimmtheitsmaß $B < 0,5$ wurde aus Platzgründen verzichtet.

Tab. 10.32: Überprüfung nicht-signifikanter Ergebnisse der Rollenanalyse

Lfd.Nr.	Rolle	Code	Variante	r	B	N
1	PB	16	$C^d_{i,k}$	0,9552	0,9125	5
2	*AR*	*0*	$C^d_{i,k}$	*-0,9357*	*0,8755*	*7*
3	AR	19	$C^d_{i,k}$	0,9327	0,8700	5
4	UM	14	$C^f_{i,k}$	-0,9299	0,8647	5
5	*BR*	*33*	$C^b_{i,k}$	*0,926*	*0,8575*	*7*
6	MD	14	$C^f_{i,k}$	0,9217	0,8495	5
7	AR	19	$C^f_{i,k}$	-0,9194	0,8454	5
8	*AR*	*0*	$C^f_{i,k}$	*0,9131*	*0,8337*	*7*
9	*AR*	*0*	$C^b_{i,k}$	*0,9083*	*0,8250*	*7*
10	FR	16	$C^b_{i,k}$	0,9004	0,8108	5
11	AR	8	$C^f_{i,k}$	-0,8888	0,7900	6
12	PB	22	$C^b_{i,k}$	-0,8875	0,7877	6
13	BR	32	$C^f_{i,k}$	0,8608	0,7410	7
14	SL	32	$C^b_{i,k}$	-0,8306	0,6898	7
15	PB	34	$C^d_{i,k}$	-0,8289	0,6871	7
16	AR	8	$C^b_{i,k}$	-0,8281	0,6857	6
17	PL	8	$C^b_{i,k}$	0,8269	0,6837	6
18	MD	19	$C^b_{i,k}$	0,8238	0,6787	5
19	SL	32	$C^f_{i,k}$	-0,8233	0,6778	7
20	PB	22	$C^f_{i,k}$	-0,8222	0,676	6
21	BR	32	$C^b_{i,k}$	0,8217	0,6752	7
22	UM	33	$C^b_{i,k}$	0,8173	0,6680	7
23	SL	1	$C^d_{i,k}$	-0,8159	0,6657	6
24	UM	16	$C^d_{i,k}$	0,8002	0,6404	5
25	VP	33	$C^b_{i,k}$	0,793	0,6288	7
26	MD	16	$C^b_{i,k}$	0,785	0,6162	5
27	PB	16	$C^f_{i,k}$	-0,7802	0,6088	5
28	BR	33	$C^f_{i,k}$	0,78	0,6084	7
29	FR	16	$C^f_{i,k}$	0,7659	0,5866	5
30	VP	22	$C^f_{i,k}$	0,7596	0,5769	6
31	PB	34	$C^f_{i,k}$	0,7533	0,5675	7
32	BR	34	$C^f_{i,k}$	0,7532	0,5673	7
33	UM	34	$C^d_{i,k}$	0,7474	0,5587	7
34	*PL*	*5*	$C^f_{i,k}$	*-0,7472*	*0,5582*	*13*
35	VP	32	$C^f_{i,k}$	0,7443	0,5539	7
36	VP	32	$C^b_{i,k}$	0,7224	0,5219	7
37	AR	19	$C^b_{i,k}$	-0,7206	0,5192	5
38	PL	8	$C^f_{i,k}$	0,7204	0,5189	6

Fortsetzung auf nächster Seite

Tab. 10.32: Überprüfung nicht-signifikanter Ergebnisse der Rollenanalyse *Forts.*

Lfd.Nr.	**Rolle**	**Code**	**Variante**	**r**	**B**	**N**
39	VP	22	$C^b_{i,k}$	-0,715	0,5112	6
40	PL	32	$C^d_{i,k}$	0,7134	0,5090	7
41	*PB*	*13*	$C^d_{i,k}$	*0,7117*	*0,5066*	*13*
42	IB	22	$C^b_{i,k}$	-0,7088	0,5024	6

Trotzdem diese Resultate – sowie die zugehörigen Koeffizienten a und b als Ergebnis der anschließenden Regressionsanalyse – aufgrund der eher kleinen Stichproben statistisch nur geringe Aussagekraft besitzen, zeigt die Tendenz, dass die berechneten Rollenausprägungen sich an die entsprechenden Fragebogenergebnisse annähern. Dies soll kurz am Beispiel des Archivars und dessen Verwendung von CLS-Code 19 illustriert werden, für den sich bei Betrachtung der bereinigten beteiligungsbezogen Sitzungsdaten der Sitzungen S_1 bis S_{27} mit einem Bestimmtheitsmaß $B^d_{9,19} = 0,3745$ (bei einem Stichprobenumfang $N = 13$) kein statistisch nachweisbarer linearer Zusammenhang ergab. Trotzdem wurden mit den Koeffizienten $a^d_{9,19} = -0,6018$ und $b^d_{9,19} = 0,0443$ anhand der entsprechenden Code-Kennzahlen $C^b_{i,k,19}$ aus den Sitzungen S_{58} bis S_{67} entsprechende Rollenausprägungen $p^d_{i,k,9,19}$ und den per Fragebogen ermittelten Rollenausprägungen $p_{k,9}$ gegenübergestellt. Das Ergebnis ist in der folgenden Tabelle 10.33 zusammengefasst.

Tab. 10.33: Evaluierung der Rollenanalyse: Rolle *AR*, Code 19, Variante (d)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,19}$	$A_{i,k}$	$C^d_{i,k,19}$	$p^d_{i,k,9,19}$	$p_{k,9}$	Δ_{abs}
1	S_{59}	T_{286}	1	1886	0,053	14,77	15	0,23
2	S_{58}	T_{291}	1	1247	0,0802	15,38	16	0,62
3	S_{60}	T_{310}	2	2498	0,0801	15,38	16	0,62
4	S_{61}	T_{302}	2	1078	0,1855	17,76	17	0,76
5	S_{65}	T_{315}	2	1066	0,1876	17,8	18	0,2

Zwar ergeben sich auch hier Abweichungen zwischen $p^d_{i,k,9,19}$ und $p_{k,9}$, diese fallen jedoch mit Werten zwischen 0,2 und 0,76 Punkten im Vergleich zu anderen, bereits zuvor diskutierten Ergebnissen gering aus, sodass das Resultat eine durchaus akzeptable Näherung darstellt, wie in der Abbildung 10.9 deutlich zu sehen ist.

Somit ergibt sich für die betrachtete Stichprobe ein Zusammenhang zwischen der Ausprägung der Rolle *Archivar* und der Verwendung von CLS++-Code 19 bei der Untersuchung der bereinigten, beteiligungsbezogenen Sitzungsdaten: Tendenziell steigt mit zunehmende Ausprägung der *Archivar*-Rolle auch die Verwendung von Code 19 innerhalb von VitaminL-Sitzungen. Obwohl dieses Ergebnis aufgrund des Stichprobenumfangs von $N = 5$ statistisch nicht zweifelsfrei belegbar, zeigt es dennoch einen Trend auf, der in möglichen Anschlussuntersuchungen als Ausgangspunkt für eben diese Untersuchungen dienen kann.

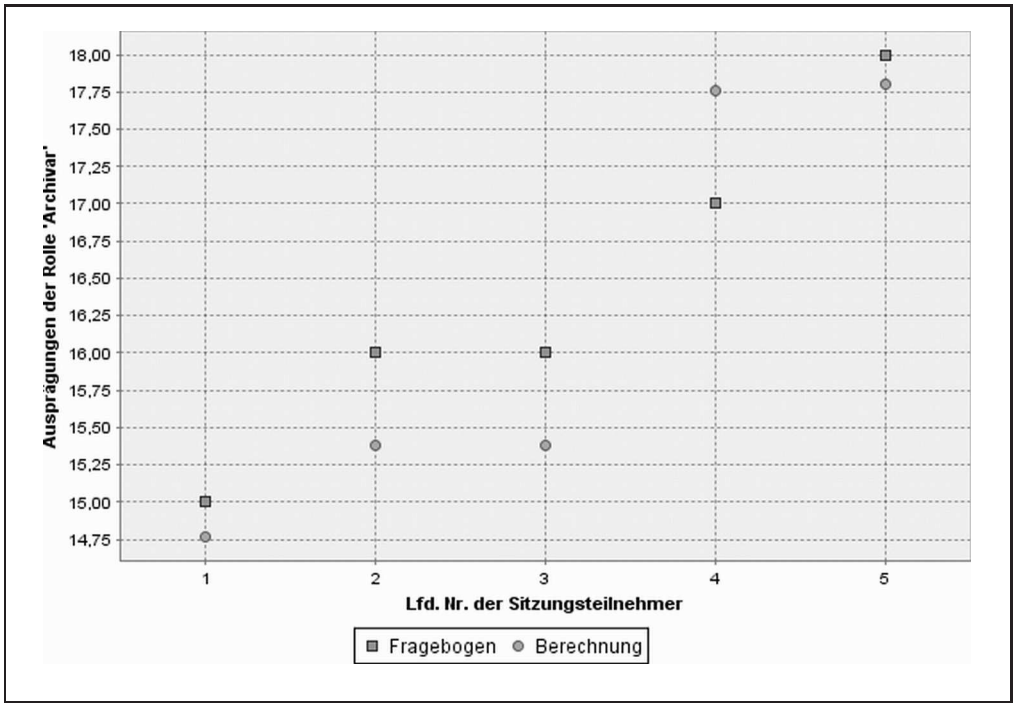

Abb. 10.9: Rollenanalyse (S_{58}-S_{67}): Rolle *AR*, Code 19, Variante (d)

Insgesamt muss festgestellt werden, dass die Untersuchungsergebnisse auf Differenzen zwischen dem Verhalten der Teilnehmer, das als Ergebnis von Benutzertests in den zugehörigen Protokolldateien aufgezeichnet wurde, und deren Aussagen, die sich durch Beantwortung des Rollenfragebogens ergeben, hinweisen.

10.5 Zusammenfassung der Rollenanalyse

Zunächst ist festzustellen, dass die aus den Code-Kennzahlen der Sitzungen S_1 bis S_{27} gewonnenen Ergebnisse (s. Tab.10.10) anhand der zu Überprüfung durchgeführten Benutzertests (Sitzungen S_{58} bis S_{67}) in ihrer ursprünglichen Form nicht bestätigt werden konnten: Die per prototypischer Analysekomponente berechneten Rollenausprägungen differieren zu sehr von den vorgegebenen Ausprägungswerten, als dass sie im Sinne einer Näherung Verwendung finden können. Vielmehr wurde aufgezeigt, dass diese zur Ergebnisevaluierung duchgeführten Sitzungen – bei gleichen Umgebungsparametern – gänzlich andere signifikante Resultate lieferten (s. Tab.10.31). Hier besteht Klärungsbedarf hinsichtlich möglicher, bislang unbekannter Einflussfaktoren, anhand derer sich die aufgetretenen Differenzen erklären lassen. Einer dieser Einflussfaktoren ist sicherlich in der groben Granularität bei der Betrachtung von Sitzungen zu suchen: In den durchgeführten Untersuchungen wurden die Benutzertests stets einer gesamtheitlichen Betrachtung unterzogen. Unter der Annahme, dass der Ablauf eines Benutzertests durch das in Kapitel 9.1 vorgestellte Vorgehensmodell einen strukturellen Rahmen besitzt, sind weiterführende Betrachtungen denkbar, bei denen die zu untersuchenden Sitzungen differenziert gemäß den Elementen des Vorgehensmodells (Teilprozesse, Phasen und Schritte)

durchgeführt werden. In Kapitel 10.5.1.3 wird dieser Gedanke aufgegriffen und näher erläutert.

Zusammenfassend kann festgestellt werden, dass sich einzelne Ergebnisse identifizieren lassen, die zwar kein direktes Näherungsverfahren darstellen, aber diskussionswürdige Ansatzpunkte bieten und im Rahmen möglicher Erweiterungen und Verfeinerungen eingehender zu betrachten sind.

10.5.1 Weiterentwicklungspotential

10.5.1.1 Kategorisierung von Rollenausprägungen

Wie bereits im Zuge der Ergebnisse des Beraters und dessen Verwendung von CLS-Code 32 angedeutet wurde (s. Kap.10.4.1.3.2 und Abbildung 10.6), kann der Übergang von einer mehrstufigen Skala (hier: 0 bis 30 Punkte) zu einer (gröberen) Ordinalskala positive Auswirkungen hinsichtlich der Aussagekraft von Ergebnissen zur Folge haben. Im genannten Beispiel würde die bereits angedachte Kategorisierung in *niedrige Ausprägung* und *hohe Ausprägung* bei einem zwischen 20 und 24 Punkten anzusiedelnden Grenzwert zwar zu einer starken Vergröberung der Resultate führen, gleichzeitig jedoch auch die Fehlerrate deutlich senken.

Betrachtet man die in der Tabelle 10.17 aufgeführten Ergebnisse, die aus der Überprüfung der Analysekomponente resultieren, so erkennt man, dass sämtliche 7 Ergebnisse $p^f_{i,k,3,32}$ der Stichprobe mit – wenn zum Teil auch nur geringen – Abweichungen gegenüber den Rollenausprägungen $p_{k,3}$ aus den Fragebögen behaftet sind. Die gesamte Stichprobe weist einen durchschnittlichen Fehler von 1,71 Punkten auf.

Nach erfolgter Einteilung in *niedrige Ausprägung* und *hohe Ausprägung* ist nur noch 1 der 7 Ergebnisse fehlerhaft: Der Teilnehmer T_{293} hat laut Fragebogen einen Ausprägungswert $p_{293,3} = 20$ hinsichtlich der Berater-Rolle erzielt und ist demzufolge mit *niedriger Ausprägung* einzustufen. Aus der Teilnahme an Sitzung S_{58} und den zugehörigen Sitzungsdaten ergibt sich für den Teilnehmer T_{293} die bereinigte zeitbezogene Code-Kennzahl $C^f_{58,293,32} = 1,0638$, anhand derer in der Analysekomponente schließlich die Rollenausprägung $p^f_{58,293,3,32} = 23,8$ berechnet wird, die wiederum fälschlicherweise zu einer Einordnung mit *hoher Ausprägung* führt. Die Abbildung 10.10 illustriert diesen Effekt.

In der Abbildung 10.10 deutlich zu sehen ist der dunkel schraffierte Bereich, in welchem ein geeigneter Grenzwert – in Gestalt einer horizontalen Linie – festzulegen ist: Alle Werte oberhalb des Grenzwerts sind als hohe Beraterausprägungen zu bewerten, alle Werte unterhalb sind niedrige Ausprägungen. Der sich aufgrund der Berechnung in der Analysekomponente als fehlerhaft erwiesene Wert ist mit einem Kreis markiert und mit einem Verweis (in Pfeilform) auf den zugehörigen Fragebogenwert versehen.

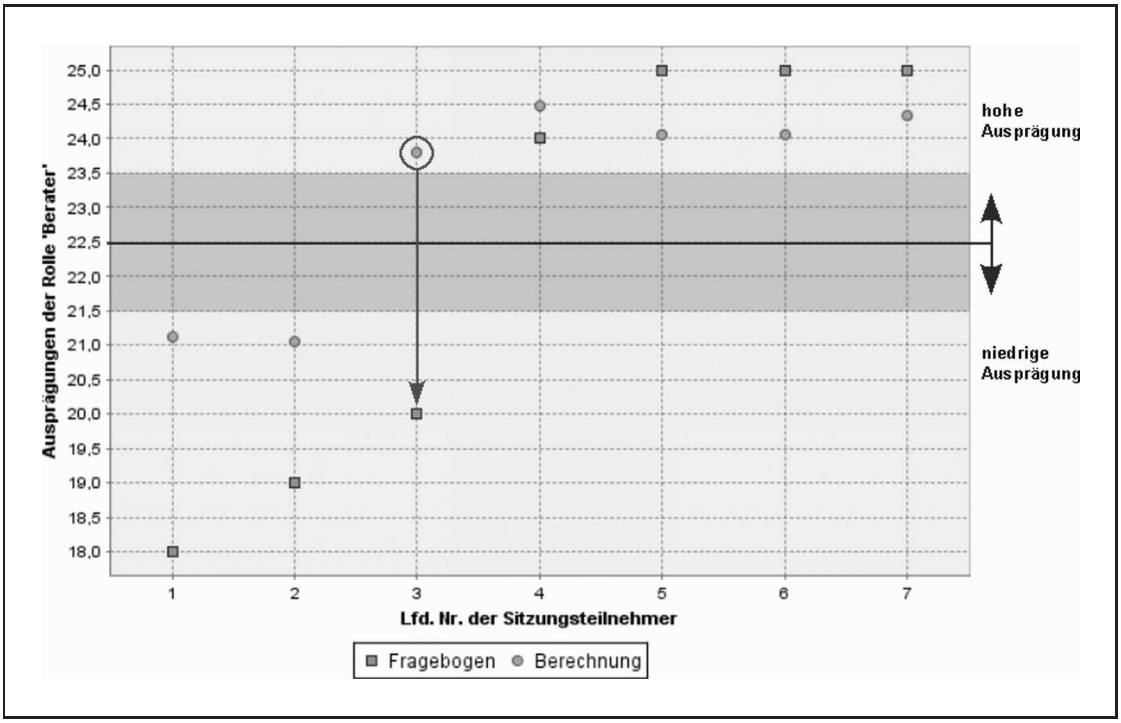

Abb. 10.10: Kategorisierug von Rollenausprägungen: Rolle *BR*

Ein weiteres Beispiel für die Anwendbarkeit dieser Vorgehensweise ergibt sich aus der Verwendung von CLS-Code 28 durch den Fragesteller, sowohl in Form der bereinigten kumulierten Code-Kennzahl $C^b_{i,k,28}$ als auch als bereinigte zeitbezogene Code-Kennzahl $C^f_{i,k,28}$. Tabelle 10.34 beinhaltet die Ergebnisse, die im Rahmen der Evaluierung der Rollenanalysekomponente berechnet wurden.

Tab. 10.34: Evaluierung der Rollenanalyse: Rolle *FR*, Code 28, Variante (f)

Lfd.Nr.	S_i	T_k	$C^a_{i,k,28}$	t_i	$C^f_{i,k,28}$	$p^f_{i,k,7,28}$	$p_{k,7}$	Δ_{abs}
1	S_{59}	T_{286}	2	102	1,9608	18,58	15	3,58
2	S_{58}	T_{293}	2	94	2,1277	18,66	16	2,66
3	S_{66}	T_{304}	1	103	0,9709	18,13	16	2,13
4	S_{59}	T_{285}	3	102	2,9412	19,03	17	2,03
5	S_{66}	T_{305}	1	103	0,9709	18,13	18	0,13
6	S_{64}	T_{294}	2	105	1,9048	18,55	19	0,45
7	S_{64}	T_{296}	2	105	1,9048	18,55	19	0,45
8	S_{63}	T_{311}	6	118	5,0847	20,01	21	0,99
9	S_{60}	T_{309}	1	112	0,8929	18,09	22	3,91
10	S_{65}	T_{315}	1	76	1,3158	18,29	22	3,71
11	S_{58}	T_{291}	1	94	1,0638	18,17	23	4,83
12	S_{66}	T_{303}	1	103	0,9709	18,13	23	4,87
13	S_{59}	T_{287}	1	102	0,9804	18,13	24	5,87

Wie der Tabelle 10.34 zu entnehmen ist, liegen alle berechneten Rollenausprägungen in einem relativ kleinen Bereich zwischen 18,13 und 20,01 Punkten (Spalte $p^f_{i,k,7,28}$). Das arithmetische Mittel dieser Stichprobe beträgt $p_\Phi = 18,49$. Bei einer Segmentierung des Stichprobenraums anhand dieses Mittelwerts entfallen fünf der 13 Rollenausprägungen $p_{k,7}$ auf das untere Segment, welches eine niedrige Ausprägung repräsentiert, die übrigen acht Ausprägung sind dem oberen Segment (= hohe Ausprägung) zuzuordnen.

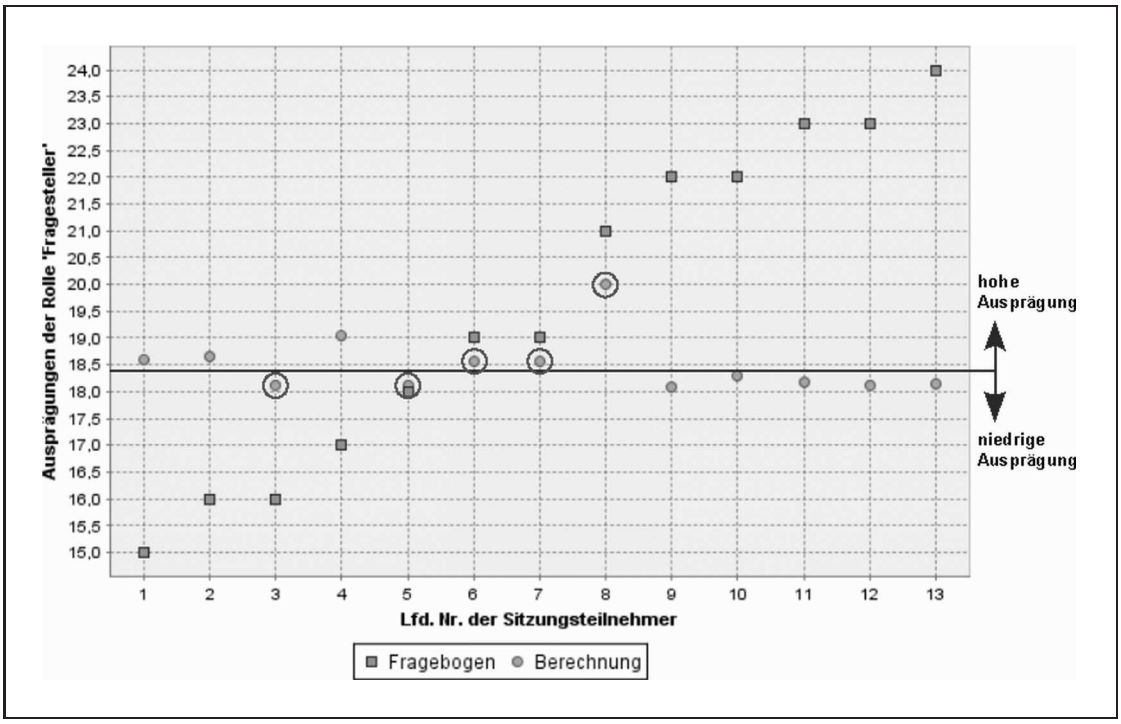

Abb. 10.11: Gegenüberstellung von Rollenausprägungen: Rolle *FR*, Code 28

Die per Fragebogen ermittelten Rollenausprägungen $p_{k,7}$ sind in dieser Abbildung als dunkle Kästchen notiert, die berechneten Ausprägungswerte $p^f_{i,k,7,28}$ der Fragesteller-Rolle sind mit kleinen Kreisen dargestellt. Wie der Abbildung 10.11 zu entnehmen ist, verhält es sich bei den zugehörigen berechneten Ausprägungswerten $p^f_{i,k,7,28}$ nahezu umgekehrt:

1. Von den acht Teilnehmern mit $p_{k,7} > p_\Phi$ wurde bei fünf Teilnehmern ein Ausprägungswert $p^f_{i,k,7,28} < p_\Phi$ berechnet, bei den anderen drei Teilnehmern ergab sich ein Ausprägungswert $p^f_{i,k,7,28} > p_\Phi$.

2. Bei den übrigen fünf Teilnehmern mit $p_{k,7} < p_\Phi$ wurde für drei Teilnehmer ein Ausprägungswert $p^f_{i,k,7,28} > p_\Phi$ und für die restlichen zwei Teilnehmer ein Ausprägungswert $p^f_{i,k,7,28} < p_\Phi$ berechnet.

Interpretiert man den Mittelwert p_Φ nun als Grenzwert, anhand dessen zwischen niedrigen und hohen Ausprägungswerten differenziert wird, und berücksichtigt man desweiteren den beobachteten Effekt zwischen den Rollenausprägungen

$p_{k,7}$ und $p^{f}_{i,k,7,28}$, so wären niedrige Ergebnisse der Analysekomponente (c) hohen Fragesteller-Ausprägungen ($p_{k,7} > p_{\Phi}$) zuzuordnen – und umgekehrt. In der vorliegenden Stichprobe wären *nur* fünf von 13 Ergebnissen falsch: zwei Teilnehmern (T_{304} und T_{305}) würde fälschlicherweise eine hohe Ausprägung zugeordnet, da bei diesen $p_{k,7} > p_{\Phi}$ gilt, bei drei anderen Teilnehmern (T_{294}, T_{296} und T_{311}) würde irrtümlich eine Einstufung in die Gruppe mit niedriger Ausprägung vorgenommen werden, denn bei diesen gilt $p_{k,7} > p_{\Phi}$.

Wie an diesen beiden Beispielen zu sehen ist, liefert die statistische Betrachtung der erfassten Sitzungsdaten und Rollenprofile über die in Kapitel 10.1 dargelegte Vorgehensweise hinaus weitere Ansatzpunkte, um Zusammenhänge zwischen Sitzungsdaten und Rollenprofilen zu identifizieren, anhand derer eine Komponente zur Rollenanalyse entwickelt werden kann.

10.5.1.2 Gruppierung von CLS-Codes

In der vorliegenden Untersuchung wurde stets die Verwendung einzelner CLS-Codes auf lineare Zusammenhänge zu Rollenausprägungen hin betrachtet. Als Ergänzung zu dieser Betrachtungsweise sind weitere Analysen unter Einbeziehung mehrerer CLS-Codes denkbar. Bereits bei den in Kapitel 10.2.4 zusammengefassten Ergebnisse, der Rollenanalyse (s. Tab.10.10) konnte festgestellt werden, dass bei einigen Rollen mehrere signifikante Ergebnisse derselben CLS++-Kategorie (s. Tab.8.4) zuzuordnen sind.

Eine Option im Rahmen solch einer aggregierten Betrachtungsweise besteht darin, die Sitzungsdaten kumuliert nach CLS-Kategorien gemäß der in Kapitel 10.1 ausgearbeiteten Vorgehensweise zu untersuchen. Auch dabei ist eine Differenzierung entsprechend der 6 verschiedenen Varianten (von *kumuliert* bis *bereinigt zeitbezogen*) zu empfehlen. Die aus diesem Ansatz hervorgehenden Ergebnisse geben Auskunft darüber, inwiefern (lineare) Zusammenhänge zwischen Rollenausprägungen und der Verwendung bestimmter CLS-Kategorien existieren.

Ein artverwandter Ansatz, bei welchem ebenfalls zwischen einzelnen CLS-Kategorien unterschieden wird, besteht in der Anwendung der multiplen Korrelation: *„Multiple Korrelation (mehrfache Korrelation) ist die Abhängigkeit einer Zufallsvariablen von zwei oder mehreren anderen im Beziehungsgefüge wirkenden Zufallsvariablen"* (Röhr et al., 1983, S.207). Für das vorliegende Problem bedeutet dies eine Klärung der Frage, ob und in welchem Maße eine Zufallsvariable X zugleich von den Zufallsvariablen $Y_1, \ldots, Y_m$ ($m \geq 2$) abhängt, wobei gilt:

- Die Variable X entpricht der Rollenausprägung $p_{k,n}$ eines Teilnehmers T_k bezüglich einer Rolle R_n.
- Die Variablen $Y_1, \ldots, Y_m$ sind Code-Kennzahlen $C^{x}_{i_1,k,l}, \ldots, C^{x}_{i_m,k,l}$, die derselben CLS++-Kategorie zuzuordnen sind.

In Kombination mit einer entsprechenden Regressionsanalyse besteht das Ziel solch einer Untersuchung darin, Funktionen der Form

$$X = a + b_1 Y_1 + \ldots + b_m Y_m \tag{10.45}$$

zu bestimmen.

10.5.1.3 Berücksichtigung des Vorgehensmodells

Sämtliche bislang durchgeführten beziehungsweise diskutierten Untersuchungen basieren auf der ganzheitlichen Betrachtung von Sitzungen und den CLS-Codes, die währenddessen von den Teilnehmern ausgelöst werden. Als Ergänzung zu dieser eher globalen und damit auch grobkörnigen Sichtweise ist eine erneute Durchführung der in diesem Kapitel beschriebenen Rollenanalyse denkbar, die das in Kapitel 9.1 ausgearbeitete Vorgehensmodell aufgreift und an die dort definierten Abschnitte – in Form von Prozessen, Teilprozessen, Phasen und Schritten – angepasst ist. Es handelt sich folglich nicht um einen neuen Ansatz, sondern um eine Anwendung des vorgestellten Verfahrens auf kleinere Sitzungsabschnitte.

Wie der Tabelle 9.3 zu entnehmen ist, sind nicht alle Rollen zu gleichen Teilen an den einzelnen Abschnitten des Vorgehensmodells beteiligt: Für die Phase *A: Problemerkennung* sind beispielsweise nur der Informationsbeschaffer, der Berater und der Planer relevant, infolgedessen wiegen erkannte Defizite hinsichtlich dieser Rollen sicherlich schwerer als eine mögliche Unterbesetzung der Umsetzer-Rolle, die im Regelfall während dieser Phase nicht zum Einsatz gelangt. Entsprechendes kann für die übrigen Phasen des fachlichen Prozesses angenommen werden. Es sei an dieser Stelle darauf hingewiesen, dass sich die Rollen des VitaminL-Rollenmodells ergänzen und insgesamt alle notwendigen Teamfunktionen abdecken, ohne sich jedoch gegenseitig auszuschließen. So können beispielsweise der Informationsbeschaffer wie auch der Archivar gleichsam zur Beseitigung von Informationsdefiziten beitragen. So betonen auch Spencer & Pruss (1995), *„dass Menschen keine Maschinen sind, die man auf diese oder jene Funktion einstellen kann; viel eher passiert es in der Praxis, dass Leute verfügbare Rollen einnehmen, obwohl sie nicht direkt auf sie zutreffen"* (Spencer & Pruss, 1995, S.91). Ähnliche Aussagen sind auch den Rollenmodellen anderer Autoren entnehmbar (s. Kap.2.5). Insofern sind auch die in vorliegenden Arbeit untersuchten Rollenausprägungen nicht als Wahrnehmung oder Nichtwahrnehmung einer Rolle zu verstehen, sondern eher als Tendenz eines Teilnehmers, eine entsprechende Rolle mehr oder weniger auszufüllen.

Um nun die Rollenanalyse so zu gestalten – und damit auch zu verfeinern –, dass nur die für die jeweilige Phase relevanten Rollen betrachtet werden, wird es zunächst erforderlich sein, die mittels der VitaminL-Software stattfindenden Sitzungen gemäß dem in Kapitel 9.1 elaborierten Vorgehensmodell zu strukturieren: Dies kann geschehen, indem sich zum Beispiel die Teilnehmer einer Sitzung mittels spezieller Bedienelemente darauf einigen, in welcher Phase sie sich jeweils befinden. Eine weitere Möglichkeit besteht darin, das System selbst entscheiden zu lassen, in welcher Phase sich ein Team zu einem gegebenen Zeitpunkt befindet.

Auch hier sind mehrere Varianten denkbar: Entscheidungen zum Wechsel in eine andere Phase werden zum Beispiel anhand der erfassten und analysierten Benutzeraktionen getroffen oder aber erfolgen beispielsweise anhand eines vorgegebenen Zeitplans, der die Dauer jeder Phase explizit festlegt. Auch ist es denkbar, dass das System den Wechsel nicht selbst durchführt, sondern nur durch Vorschlagen bei den Teilnehmern anregt. Mit solch einer Strukturierung einhergehen kann eine mögliche Anpassung der Software – und vor allem der Schnittstelle zum Benutzer – entsprechend der jeweiligen Phase: Unter Umständen werden in einigen Phasen zum Beispiel keine Quelltext-Dokumente benötigt, sofern in solch einer Phase die Teilnehmer ausschließlich untereinander kommunizieren. Ein ähnliches Konzept strukturierter Zusammenarbeit wird übrigens auch in der Arbeit von McManus & Aiken (1995) verwendet (s. Kap.3.1.4.1).

Unabhängig davon, wie solch eine Strukturierung innerhalb der VitaminL-Software nun tatsächlich ausgestaltet würde, kann die Rollenanalyse an die diversen Phasen des Vorgehensmodells angepasst werden. Anstelle eines globalen Analyseverfahrens (pro Rolle) ergeben sich von Phase zu Phase lokale Analyseverfahren, die sich zudem deutlich voneinander unterscheiden können, was bedeuten kann:

- Es kommt in allen Phasen derselbe Algorithmus zum Einsatz, der jedoch von Phase zu Phase mit jeweils anderen Parametern aufgerufen wird.
- Für jede Phase wird ein anderer Algorithmus formuliert und verwendet.

Mischformen dieser beiden Ansätze sind ebenfalls möglich.

Die Rollenanalyse innerhalb von Aktivitäten des sozialen Prozessen erfordert keine gesonderte Betrachtung: Zwar handelt es sich hierbei gemäß den Ausführungen aus Kapitel 9.1.2 stets um eine Störung des normalen, aufgabenorientierten Ablaufs, bei denen eine korrigierendes Eingreifen vor allem dann notwendig ist, wenn die Störung länger anhält, jedoch kann eine Entscheidung, ob sich die Teilnehmer einer Sitzung gerade innerhalb einer sozialen Aktivität befinden, analog zur Phasenfestlegung entweder durch die Teilnehmer oder das VitaminL-System erfolgen.

Ob mittels einer Berücksichtigung des Vorgehensmodells bei entsprechender Strukturierung der VitaminL-Sitzungen tatsächlich differenziertere Ergebnisse hinsichtlich der Rollenanalyse erzielt werden können, muss in Forschungen, die sich an die vorliegende Arbeit anschließen, erst noch gezeigt werden.

10.5.1.4 Weitere Ansätze

10.5.1.4.1 Nicht-lineare Regression Im Rahmen weiterer Untersuchungen zur Aufdeckung möglicher Zusammenhänge der Form $y = f(x)$ zwischen Rollenausprägungen und Benutzeraktionen (in Form von CLS-Codes) ist ein Übergang von der linearen Regression zur nicht-linearen Regression denkbar. In der Literatur (vgl. Röhr et al., 1983, Kap.8.5; Clauss et al., 1995, S.85) werden als häufig verwendete Ansätze für nichtlineare Funktionen unter anderem die folgenden Funktionen genannt:

- Quadratische Funktion: $f(x) = a + bx + cx^2$
- Exponentialfunktion: $f(x) = ae^{bx}$
- Logarithmusfunktion: $f(x) = a + b \ln x$
- Logistische Funktion $f(x) = \frac{a}{1+be^{-cx}}$

Üblicherweise geht man von der Darstellung einer Stichprobe als Punktwolke aus und wählt denjenigen Ansatz, der der in der Punktwolke vermuteten Kurvenform an nächsten kommt. Diese Arbeit erledigen heutzutage auch gängige Statistik-Pakete auf dem Computer.

10.5.1.4.2 Verfahren des Data Mining und der KI Neben der bereits ausführlich behandelten Regressionsanalyse ist der Einsatz gängiger Verfahren denkbar, die der KI oder dem Data Mining zugeordnet werden. Im Bereich der KI anzusiedeln sind beispielsweise *Künstliche Neuronale Netze*, Bayessche Netzwerke oder auch Hidden-Markov-Modelle, um nur einige der Konzepte zu nennen, mit denen Programme in die Lage versetzt werden, dazu zu lernen. Für die vorliegende Problematik hieße dies, eine Analysekomponente zu entwerfen, die auf diesen Konzepten basiert und somit in die Lage versetzt wird, anhand der Eingangsdaten – in Form von Rollenausprägungen und Sitzungsdaten (resp. Code-Kennzahlen) – dazuzulernen und ihr Wissen über Zusammenhänge zwischen Rollenausprägungen und Code-Kennzahlen mit jeder stattfindenden Sitzung zu aktualisieren.

Zu den klassischen Problemtypen des Data Mining gehören unter anderem die Abhängigkeitsanalyse, zu der als Spezialfall auch die Korrelationsanalyse gezählt wird, die Cluster-Analyse, mit deren Hilfe Daten in sinnvolle Teilmengen aufgespalten werden können, oder auch Prognoseprobleme, die anhand eines gegebenen Input den zu erwartenden Output bestimmen. So könnte beispielsweise die Cluster-Analyse zum Einsatz kommen, um – wie in Kapitel 10.5.1.2 vorgeschlagen, Gruppen von CLS-Codes auf Abhängigkeiten hin zu untersuchen.

10.5.1.4.3 Inhaltsanalyse Die Abbildung sämtlicher Benutzerinteraktionen auf die in Kapitel 8.4.3.2 definierten CLS++-Codes stellt zweifelsfrei eine Reduktion dieser Interaktionen dar, die jedoch zugunsten der Verarbeitungsgeschwindigkeit zunächst in Kauf genommen wurde. Trotzdem sind auch Analyseverfahren denkbar, die über die reine Analyse auf der Basis von ganzen Zahlen (in Form der CLS-Codes) hinausgehen und die Zusatzinhalte von Benutzerinteraktionen – wie zum Beispiel die Textinhalte von Kommunikationsbeiträgen – berücksichtigen.

10.5.2 Fazit

Die Korrelationsanalyse – und damit verbunden auch die einfache lineare Regressionsanalyse – stellt in Verbindung mit den erzielten Ergebnissen einen ersten An-

satz auf dem Weg zu einer Software-Komponente dar, die innerhalb eines virtuellen Tutors eingesetz wird, um während einer Sitzung die Frage nach dem Grad der Rollenzugehörigkeit der einzelnen Teilnehmer so zu beantworten, dass bei Bedarf Unterstützungsangebote durch den virtuellen Tutor generiert werden können, die an die Rollenzusammensetzung des jeweiligen Teams angepasst sind und etwaige Defizite hinsichtlich benötigter Rollen kompensieren. Wie in den abschließenden Ausführungen aufgezeigt werden konnte, stellt diese Form der Analyse nicht den einzigen Ansatz dar, mit dem Zusammenhänge zwischen Rollenausprägungen und Benutzeraktionen untersucht und gegebenenfalls auch identifiziert werden können. Vielmehr ist die Kombination aus Korrelationsanalyse und linearer Regressionsanalyse als wichtiger Ausgangspunkt zu verstehen, auf dem basierend ein Ausbau des Prototypen zu der geforderten Rollenanalysekomponente stattfinden kann. Im Zuge dieses Ausbaus stellt die Korrelationsanalyse nur einen Baustein dar, der in einer Art Methodenmix sowohl mit den im Ausblick vorgestellten Ansätzen, aber möglicherwiese auch mit weiteren, noch zu ermittelnden Konzepten zu kombinieren ist.

Kapitel 11

Analyse von Problemsituationen

Unter dem Begriff *Problem* werden im Folgenden sowohl allgemeine Schwierigkeiten, die die Aufgabenbearbeitung erschweren oder behindern, als auch konkrete Fehler, die beispielsweise aus Übersetzungsversuchen syntaktisch inkorrekter Quelltexte resultieren, verstanden.

Während die Rollenanalyse des vorigen Kapitels entsprechend des Schwerpunkts der vorliegenden Arbeit detailliert durchgeführt wurde und mittels prototypischen Software-Komponenten überprüft werden konnte, wird die Problemanalyse aus eher konzeptioneller Sicht betrieben. Die Resultate dieser Analyse liegen daher auch eher in Form von Maßnahmen und Konzeptvorschlägen vor als in fertigen Software-Komponenten. Die Grundlage für die Analyse von Problemsituationen bilden Benutzertests, deren Protokolldateien mit intellektuellem Aufwand analysiert werden (s. Kap.11.2). Diese werden ergänzt um Beobachtungen der Teilnehmer durch Tutoren während der Benutzertests und während der gemeinsamen Bearbeitung von Beispielen (s. Kap.7.1.2.1) und Übungsaufgaben (s. Kap.7.1.2.2). Ferner wurden Teilnehmer nach ihren persönlichen Problemerfahrungen befragt (s. Kap.11.1.3).

11.1 Problemklassifikation

Die Klassifikation der denkbaren Problemsituationen folgt dem in Kapitel 9.1 festgelegten Vorgehensmodell, wobei eine Differenzierung zwischen fachlichen und sozialen Problemen erfolgt.

11.1.1 Fachliche Problemsituationen

Die fachlichen Problemsituationen ergeben sich unmittelbar aus den Teilprozessen, Phasen und Schritten des fachlichen Prozesses (s. Kap.9.1.1) und lassen sich gemäß der vereinbarten Teilprozesse weiter gliedern. Die einzelnen Schritte als

kleinste Einheit des Vorgehensmodells werden dabei nicht nur als zu durchlaufende Elemente bei der Aufgabenbearbeitung, sondern zugleich auch als potentielle Problemquellen angesehen.

11.1.1.1 Probleme der Arbeitsvorbereitung

Der Teilprozess *Arbeitsvorbereitung* besteht aus den drei Phasen *Problemerkennung*, *Herstellung des Lernkontexts* und *Organisation der Problemlösung* mit insgesamt Phasen zugehörigen Schritten (s. Tab.9.3). Es ergeben sich folgende mögliche Kategorien von Problemsituationen, die sowohl einzelne Mitglieder wie auch das gesamte Team betreffen können, wobei die Anfangsbuchtaben der einzelnen Kategorien hier und im Folgenden mit den entsprechenden Phasen des Vorgehensmodells (s. Kap.9.3.1, Tab.9.3) korrespondieren:

A.1 Aufgabenstellung unklar:
Das Lesen der Aufgabenstellung bereitet insofern Schwierigkeiten, als dass anschließend keine Klarheit hinsichtlich der Aufgabe selber besteht.

A.2 Kernproblem unbekannt:
Das aus der Aufgabenstellung resultierende Ziel wurde erkannt, das darin enthaltene Kernproblem konnte jedoch nicht identifiziert werden.

A.3 Teilprobleme nicht erkennbar:
Es herrscht Konsens bezüglich des zu lösenden Kernproblems, enthaltene Teilprobleme wurden jedoch (in Teilen oder als Ganzes) noch nicht identifiziert.

B.1 Keine Zuordnung von Lerninhalten zu Teilproblemen möglich:
Sowohl das Kernproblem als auch dessen Teilprobleme wurden erkannt, es bereitet dem Team jedoch Schwierigkeiten, zugehörige Lerninhalte zu identifizieren und zuzuordnen.

B.2 Lösungsansätze in Lerninhalten nicht ermittelbar:
Trotz Identifikation der für die Aufgabenlösung relevanten Lerninhalte gelingt es nicht, passende Lösungsansätze zu eruieren.

B.3 Bewertung von Lösungsansätzen problematisch:
Das Team hat Probleme, die einzelnen Lösungsansätze hinsichtlich ihrer Eignung zu bewerten, wobei die Aufgabenstellung wie auch aus ihr resultierende Anforderungen zu berücksichtigen sind.

B.4 Auswahl geeigneter Lösungsansätze problematisch:
Die Auswahl derjenigen Lösungsansätze, die im Hinblick auf die nachfolgende Umsetzung am besten geeignet erscheinen, ist mit Schwierigkeiten behaftet.

C.1 Definition von Teilaufgaben problematisch:
Obwohl sich die Gruppe auf alle für die Umsetzung benötigten Lösungskonzepte und -ansätze einigen konnte, lassen sich diese nicht auf zu bearbeitende Teilaufgaben abbilden.

C.2 Verteilung der Aufgaben problematisch:
Das Team ist nicht in der Lage, die für eine erfolgreiche Aufgabenbearbeitung zu erledigenden (Teil-)Aufgaben geeignet auf die einzelnen Teammitglieder zu verteilen.

Wie bereits in Kapitel 9.1.1 festgestellt wurde, ist auch hier anzumerken, dass ein Übergang zu einer bestimmten Einheit des Vorgehensmodells erst dann sinnvoll ist, wenn alle vorherigen Einheiten erfolgreich abgeschlossen werden konnten. Dies kann aber auch bedeuten, dass im Bedarfsfall ein Rücksprung zu einer vorgelagerten Einheit durchgeführt werden kann. Dies kann beispielsweise notwendig sein, wenn bei auftretenden Problemen keine Lösung gefunden werden kann, so dass eine Revision von bisherigen Lösungsansatzen und/oder Rahmenbedingungen ermöglicht wird.

11.1.1.2 Probleme bei der Umsetzung

Die Umsetzung nach erfolgter Arbeitsvorbereitung (inkl. Aufgabenverteilung) beinhaltet die eigentliche Programmierung, bei der üblicherweise jedes Teammitglied *seine* Quelltexte erstellt und bearbeitet, diese jedoch bei Bedarf den anderen Teammitgliedern zur weiteren Bearbeitung, beispielsweise zur Korrektur einzelner Passagen, verfügbare machen kann. Gemäß dem Vorgehensmodell erfolgt eine Differenzierung nach *Codierung der Rohfassung*, *Fehlerbereinigung* und *Abschlussarbeiten*, wobei die letztgenannte Phase sich bei Beobachtungen als weitestgehend unproblematisch erwiesen hat: Der intellektuelle Aufwand für das korrekte Formatieren von Quelltexten sowie das Verfassen von Kommentaren zur Dokumentation von Quelltexten ist als eher gering einzustufen; Probleme in dieser Phase treten – wenn überhaupt – bei Teilnehmern auf, deren Muttersprache nicht deutsch ist.

11.1.1.2.1 Probleme bei Codierung der Rohfassung Die Rohfassung entspricht einer prototypischen Version des fertigen Quelltexts. Auf dem Weg zur Rohfassung können folgende Problemsituationen festgestellt werden:

D.1 Suche nach richtigem Template problematisch:
Teilnehmer haben Probleme, passende Code-Beispiele als Ausgangspunkt ihrer Rohfassung zu finden.

D.2 Schwierigkeiten bei Anpassung eines Templates:
Die Adaption eines gewählten Code-Beispiels an das aktuelle (Teil-)Problem erweist sich als problematisch.

D.3 Probleme mit Integration von Templates in Gesamtlösung:
Das Zusammensetzen der einzelnen Templates zu einer Gesamtlösung bereitet Probleme.

Die Rohfassung ist erwartungsgemäß fehlerbehaftet, was an sich noch kein Problem darstellt, solange die enthaltenen Fehler identifiziert und bereinigt werden können.

11.1.1.2.2 Probleme der Fehlerbereinigung Die Beseitigung von Fehlern geschieht üblicherweise in einem iterativen Prozess, bei dem – oft mehrmals – der Code-and-fix-Zyklus (s. Abb.9.1) durchlaufen wird.

Prozessorientierte Sichtweise Im Zuge einer Fehlerbereinigung nach dem an das Vorgehensmodell adaptierten Code-and-fix-Zyklus können folgende Problemsituationen beobachtet werden:

E.1 Übersetzung mit Compiler problematisch:
Ein Teilnehmer hat Schwierigkeiten, einen Quelltext per Java-Compiler übersetzen zu lassen.

E.2 Erkennen von Compile-Fehlern problematisch:
Es wird nicht erkannt, ob beziehungsweise dass beim (korrekten) Aufruf des Compilers mit einer zu übersetzenden Quelltext-Datei der Übersetzungsvorgang Fehler im Quelltext identifiziert hat.

E.3 Problem beim Lokalisieren von Compile-Fehlern:
Teilnehmer erkennen zwar, dass ein Übersetzungsvorgang mit Fehlermeldungen endet, können diese aber nicht den entsprechenden Stellen im Quelltext zuordnen.

E.4 Problem beim Finden einer Maßnahme zur Beseitigung von Compile-Fehlern:
Obwohl die Stelle im Quelltext identifiziert werden konnte, der der Compiler einen Fehler zugeordnet hat, bereitet es den Teilnehmern Schwierigkeiten, eine geeignete Maßnahme zur Beseitigung des Fehlers zu festzumachen.

E.5 Schwierigkeiten bei der Anwendung von Fehlerbeseitigungsmaßnahmen:
Es gibt für einen erkannten und lokalisierten Fehler eine geeignete Maßnahme zu dessen Beseitigung, jedoch bereitet es den Teilnehmern Probleme, diese auch umzusetzen.

E.6 Probleme beim Ausführen mit dem Interpreter

E.7 Kein Erkennen von Laufzeitfehlern

E.8 Lokalisierung von Laufzeitfehlern nicht möglich

E.9 Schwierigkeiten bei Identifizierung von Maßnahmen zur Beseitigung von Laufzeitfehlern

E.10 Probleme bei der Anwendung von Maßnahmen zur Beseitigung von Laufzeitfehlern

Die Punkte (E.6) bis (E.10) sind analog zu (E.1) bis (E.5) zu betrachten, jedoch bezogen auf den Interpreter und Laufzeitfehler anstelle von Compiler und Übersetzungsfehler.

Fehlerorientierte Sichtweise In diesem Zusammenhang ist auch eine weitere Sichtweise denkbar, die die eigentlichen Programmierfehler – als Ergebnis der Programmierung – fokussiert: In Anlehnung an die in BISCHOFF (2005) festgelegte Klassifikation (BISCHOFF, 2005, S.65ff) sowie unter Einbeziehung der eigenen Ergebnisse aus der in Kapitel 7.2 dargelegten Fallstudie kann eine Differenzierung nach Syntax-, Semantik- und Logikfehlern sowie Bedienfehlern[1] eine sinnvolle Ergänzung zur eben genannten Klassifikation darstellen.

Ein Syntax-, Compiler- oder auch Übersetzungsfehler wird in dieser Arbeit verstanden als *„bereits zur Kompilierzeit an Fehlermeldungen"* angezeigter und erkennbarer Fehler (BISCHOFF, 2005, S.66). Im Rahmen ihrer Untersuchungen strukturiert BISCHOFF (2005) die von ihr identifizierten Syntax-Fehler weiter wie folgt:

Ü.1 Punkt- und Trennzeichen:
In diese Kategorie fallen sämtliche Fehler, die aus dem fehlerhaften Gebrauch (inkl. Weglassen) von Punkt- und Trennzeichen wie Kommata, Semikolons, Punkt oder auch Doppelpunkt resultieren.

Ü.2 Klammersetzung:
Hierzu zählen fehlende oder falsch verwendete runde und geschweifte Klammern. Syntaktisch korrekte Klammersetzungen, die zu fehlerhaften Berechnungen führen, werden in dieser Kategorie genausowenig erfasst wie die Verwendung von eckigen Klammern.

Ü.3 Schlüsselwörter:
Fehlerhaft geschriebene Schlüsselwörter sowie falsch eingesetzte oder ausgelassene Schlüsselwörter werden dieser Kategorie zugeordnet, sofern sie beim Compile-Vorgang entsprechende Fehlermeldungen generieren.

Ü.4 Variablen und Konstanten:
Jegliche Fehler bezüglich Definition, Deklaration, Initialisierung und Verwendung von Variablen und Konstanten fallen in diese Kategorie. Dies umfasst auch solche Fehler, die als Folge von Schreibfehlern und abweichenden Namen auftreten.

Ü.5 Datentypen und Arrays:
Diese Kategorie umfasst Fehler, die aus unzulässiger Typumwandlung (*casting*), Parameter mit falschem Datentyp bei Methodenaufrufen und Inkompatibilitäten zwischen Rückgabewerten und aufnehmenden Variablen resultieren. Ferner werden sämtliche Fehler im Zusammenhang mit Arrays wie falsche Initialisierung, fehlerhafte Array-Größe, Fehler beim Zugriff auf Array-Elemente via Index und Fehler beim Gebrauch eckiger Klammern.

Ü.6 Klassen und Methoden:
Dieser Kategorie lassen sich alle Fehler zuordnen, die aus der Deklaration,

[1] Unter Bedienfehlern werden in diesem Zusammenhang Fehlbedienungen der Werkzeuge der Programmierung wie Quelltexteditor, Compiler und Interpreter verstanden.

der Definition oder dem Aufruf von Klassen und Methoden resultieren. Dazu gehören Fehler bei der Groß- und Kleinschreibung, bei der Verwendung von Parameterlisten, beim statischen Zugriff auf Klassenelemente und die Konvention der Übereinstimmung von Klassenname und Dateiname.

Ü.7 Vererbung:
Syntaxfehler dieser Kategorie sind fehlende oder unerlaubte Ableitungen von einer Oberklasse, fehlende `import`-Anweisungen sowie unvollständige Implementierungen von Schnittstellenklassen (sog. *interfaces*).

Ü.8 Sonstige Syntax-Fehler:
Sofern vom Compiler erkannte Fehler sich keiner der vorigen Kategorien zuordnen lassen, werden sie automatisch dieser Kategorie zugeordnet.

Während die Syntax-Fehler bereits zu einem frühren Zeitpunkt auftreten und somit ebenso früh erkannt und behoben werden können, treten sowohl Semantik- wie auch Logik-Fehler erst während des Programmablaufs auf, wobei *„semantische Fehler zu Laufzeitfehlern (Exceptions) führen"* (Bischoff, 2005, S.66) und damit zu vorzeitigen, unerwünschten Programmabbrüchen führen, wohingegen Logik-Fehler sich in unerwünschten Verhalten respektive in unerwarteten, fehlerhaften Ergebnissen und Programmausgaben zeigen (vgl. Balzert, 1999, S.172).

Die Einteilung der Semantik-oder auch Laufzeit-Fehler wird wie folgt vorgenommen:

L.1 NullPointerException:
Diese Aufnahme tritt in der Regel auf bei dem Versuch, auf nicht-belegte Objektreferenzen (sog. *null*-Referenzen) zuzugreifen. Ursache sind in der Regel nicht initialisierte Objektvariablen.

L.2 ArrayIndexOutOfBoundsException:
Die Elemente eines n-elementigen Arrays können direkt mit einem Index i adressiert werden, für den $0 \leq i < n$ gilt. Der Zugriff mit anderen Indexwerten löst eine Ausnahme vom Typ ArrayIndexOutOfBoundsException aus.

L.3 ArithmeticException:
Nicht-erlaubte arithmetische Operationen wie beispielsweise eine Division durch 0 führen zu diesem Laufzeitfehler.

L.4 IOException:
Diese Ausnahme tritt im Zusammenhang mit Ein-/Ausgabe-Operationen im Allgemeinen auf und signalisiert, dass die entsprechende Operation nicht erfolgreich durchgführt werden konnte. Das Spektrum möglicher Fehler reicht von fehlgeschlagenen Versuchen, Dateien anzulegen, über Schreibfehler bis hin zu Lesefehlern.

L.5 Sonstige Abbrüche:
Mit den vier genannten Ausnahmesituationen ist zwar die Menge möglicher

Laufzeitfehler nicht annähernd erfasst[2], jedoch decken diese die von Programmierneulingen am häufigsten verursachten Laufzeitfehler ab. Weitere Ausnahmen werden daher dieser fünften Kategorie zugeordnet.

Die Kategorie der Logik- oder auch Verhaltensfehler lässt sich folgendermaßen strukturieren:

V.1 Ausgabefehler:
Unter einem Ausgabefehler wird eine falsch positionierte oder fehlende Ausgabeanweisung verstanden.

V.2 Rechenfehler:
Ein Rechenfehler entspricht einer syntaktisch korrekten Berechnung, die fehlerhafte Ergebnisse liefert.

V.3 Verzweigungsfehler:
Ein Verzweigungsfehler tritt bei `if`- und `if-else`-Anweisungen in Form von falsch formulierten Bedingungen auf.

V.4 Fallunterscheidung:
Von einer fehlerhaften Fallunterscheidung ist dann die Rede, wenn eine `switch-case`-Anweisung nicht wie gewünscht – beziehungsweise erwartet – abgearbeitet wird.

V.5 Schleifenfehler I:
Ein Schleifenfehler vom Typ I tritt immer dann in Erscheinung, wenn sich ein Programm aufgrund falscher Startwerte einer Schleife fehlerhaft verhält.

V.6 Schleifenfehler II:
Ein Schleifenfehler vom Typ II umfasst sonstiges Fehlverhalten eines Programms, das sich als Folge fehlerhaft formulierter Schleifen (wie Endlosschleifen, falsche `break`-/`continue`-Anweisungen etc.) ergibt.

V.7 Sonstiges Fehlverhalten:
In diese Kategorie fällt sonstiges Fehlverhalten eines Programms, das zur Laufzeit festgestellt werden kann, aber nicht in einem Laufzeitfehler/Programmabbruch endet.

Eine weiterführende Betrachtung von Fehlverhalten, wie sie beispielsweise von Allen (2002) mit dessen *bug patterns* durchgeführt wurde, ist in Anbetracht der Zielgruppe zunächst wenig sinnvoll: Die *bug patterns* setzen auf Designfehlern und Objektorientierung auf, während die aus den Beobachtungen von Programmieranfängern extrahierbaren Fehler zunächst der prozeduralen Programmierung anstelle der objektorientierten Programmierung zuzuordnen sind.

[2] Das API des JDK SE 5.0 benennt allein an die 70 direkte Ableitungen der allgemeinen Ausnahmeklasse `java.lang.Exception`, von denen wiederum viele selber Ableitungen besitzen.

Bedienfehler (oder auch Nutzungsfehler) repräsentieren Fehler oder Probleme beim Aufruf derjenigen Programme, die für die Entwicklung eigener Quelltexte sowie die Generierung und Ausführung der zugehörigen Anwendungen eingesetzt werden. Fehler dieser Kategorie sind stets von den verwendeten Programmen abhängig. Nachfolgende Betrachtungen basieren auf dem Einsatz der VitaminL-Software und beinhalten daher ausschließlich Fehler und Probleme, die sich in Folge der Bedienung des VitaminL-Clients ergeben:

N.1 Dokument laden/importieren:
Probleme dieser Kategorie signalisieren Schwierigkeiten beim Versuch, ein bereits auf einem Datenträger vorhandenes Dokument zur Bearbeitung in der VitaminL-IDE zu öffnen.

N.2 Dokument speichern:
Beim Speichern eines Dokuments treten Probleme auf. Hierzu gehört auch das Umbenennen eines Dokuments, also das Abspeichern unter einem anderen Namen.

N.3 Dokument bearbeiten:
Es bereitet einem Anwender Schwierigkeiten, ein bereits in der VitaminL-IDE geöffnetes Dokument zu bearbeiten. In diese Kategorie fallen auch das Anfordern und das Freigeben von Dokumenten.

N.4 Aufruf des Compilers:
Der Aufruf des Compilers zur Übersetzung eines Quelltext-Dokuments bereitet Schwierigkeiten.

N.5 Aufruf des Interpreters:
Der Aufruf des Interpreters - und damit der Versuch, das eigene Programm auszuführen - erweist sich als problematisch.

N.6 Sonstige Bedienprobleme:
Alle während des Arbeitens mit der VitaminL-IDE auftretenden Bedienprobleme, die keiner der zuvor genannten Kategorien zugeordnet werden können, sind hier zu einzuordnen.

11.1.1.2.3 Probleme der Abschlussarbeiten Der Vollständigkeit halber werden auch mögliche Probleme mit den Abschlussarbeiten in die Analyse von Problemsituationen einbezogen:

F.1 Formatierung des Quellcodes problematisch:
Es bereitet den Teilnehmern Schwierigkeiten, die Quelltexte – unter Zuhilfenahme von Leerzeichen, Leerzeilen und Zeilenenden – so zu formatieren, dass deren äußere Form den vereinbarten Anforderungen genügt.

F.2 Schwierigkeiten mit Kommentaren:
Das Ergänzen der Quelltexte um (deutschsprachige) Kommentare zur Dokumentation der Quelltexte erweist sich als problematisch.

11.1.1.3 Probleme des Unterstützungsprozesses

Der Unterstützungsprozess als Teilprozess des fachlichen Prozesses begleitet die Aktivitäten der Arbeitsvorbereitung und der Umsetzung. Die enthaltenen Phasen *Koordination*, *Informationsbeschaffung* und *Fachliche Kommunikation* gliedern sich in insgesamt neun Schritte mit entsprechendem Problempotential:

X.1 Nichteinhaltung von Fristen und Terminen

X.2 Keine Koordination von Teilaufgaben

X.3 Probleme bei der Überprüfung von Ergebnissen

Y.1 Schwierigkeiten bei der Recherche

Y.2 Nachfragen beim Tutor problematisch

Z.1 Probleme mit Fragestellungen

Z.2 Erklärungen schwierig

Z.3 Probleme mit Meinungsäußerungen

Z.4 Liefern von Ideen problematisch

Somit ergibt sich ein umfassendes Verzeichnis möglicher Problemsituationen, die prinzipiell dem fachlichen Prozess des Vorgehensmodells zuzuordnen sind und durch die Problemsituationen des sozialen Prozesses zu ergänzen sind.

11.1.2 Soziale Problemsituationen

Die Elemente des sozialen Prozesses wurden bereits in Kapitel 9.1.2 eingehend erläutert. Insbesondere soll an dieser Stelle nochmals auf den Störungscharakter der einzelnen Elemente hingewiesen werden:

S.1 Konfliktsituationen:
Auseinandersetzungen zwischen Teammitgliedern, die den fachlichen Rahmen verlassen und bisweilen auch persönliche Verbalangriffe beinhalten, fallen in diese Kategorie.

S.2 Motivationsprobleme:
Teilnehmer sind nur unzureichend motiviert und müssen immer wieder zur Teilnahme angeregt werden.

S.3 Off-Topic-Aktivitäten:
Hierzu zählen Aktivitäten von Teilnehmern – in der Regel in Form von Kommunikationsbeiträgen – , die nicht der Aufgabenbearbeitung zuzuordnen sind, sondern sich anderen Themen widmen.

S.4 Inaktivitäten:
Liefert ein Teilnehmer über einen nicht vernachlässigbaren Zeitraum keinen konstruktiven Beitrag zur Gruppenarbeit, handelt es sich um eine Inaktivität.

S.5 Kommunikationsprobleme:
Sämtliche Problemsituationen, deren Ursachen in der Kommunikation selber zu suchen sind (vgl. GÖLDNER, 2005), entfallen auf diese Problemgruppe.

Das gesamte Klassifikationsschema basiert auf Studien, die im Rahmen des VitaminL-Forschungsprojekts durch dessen Teilnehmer durchgeführt wurden. Ergänzt werden diese durch Beobachtungen und Erhebungen, die im Vorfeld des Projekts stattfanden (s. Kap.7.2).

11.1.3 Probleme aus Benutzersicht

Mit dem Ziel, das Klassifikationsschema auf Anwendbarkeit im betrachteten Kontext hin zu überprüfen, fand im Anschluss an die im Dezember 2005 durchgeführten Benutzertests (s. Kap.10.3) eine Befragung unter den Teilnehmern statt. Mit der Fragestellung *„Welche konkreten Problemsituationen bei der Java-Programmierung können Sie benennen? Mit welchen Problemsituationen wurden Sie während der Java-Programmierung bereits konfrontiert?"* wurden die Teilnehmer nach für sie typischen Problemsituationen während der Java-Programmierung befragt. Mehrfachnennungen waren erlaubt. Die Antworten der 13 Teilnehmer, die diese Fragestellung beantworteten, sind wörtlich in der folgenden Tabelle notiert.

Tab. 11.1: Problemnennungen von Teilnehmern

Lfd.Nr.	**Problembeschreibung**	**Kategorie**
1	Dass der Quelltext nicht funktioniert und man nicht weiß, warum.	E.3
2	Dass der Quelltext syntaktisch richtig ist, aber nicht macht, was man erwartet	E.8
3	Man wusste manchmal nicht wie man ein bestimmtes Problem lösen sollte und hat in den Unterlagen aus den Vorlesungen nachgeschaut. Dort waren jedoch die Probleme zu 100% gelöst, also hatte man gleich die komplette Lösung, anstatt eines hilfreichen Ansatzes (Beispiel: FileReader/Writer, BubbleSort).	B.2
4	Fehlende Sprachkenntnisse (z.B. Innere Klassen)	B.1
5	Probleme mit der API	Y.1
6	Kompilierfehler, da das Programm die Endung .java nicht automatisch setzt	Bed.2
7	Lokale Variablen wurden flüchtigerweise nicht als solche erkannt	E.3, Syn.4

Fortsetzung auf nächster Seite

Tab. 11.1: Problemnennungen von Teilnehmern *Forts.*

Lfd.Nr.	Problembeschreibung	Kategorie
8	Zuerst war es sehr schwierig mit Eclipse zu programmieren, weil ich nicht wusste, wie man damit Programme ausführen kann, aber als ich das verstanden habe, hab ich entdeckt, dass Eclipse ganz bequem ist.	Bed[3]
9	Ich konnte nicht finden, wie man Zeilenabstand bei Programmieren mit Swing machen kann	D.2
10	Es ist schwierig für mich, Kommentare zu schreiben, weil Deutsch keine Muttersprache für mich ist	F.2
11	Schon mehrmals war so, dass Programm nicht funktioniert und ich nicht weiß, wie man das Problem lösen kann	E
12	Damit, dass man kurzzeitig nicht mehr weiter wusste	U
13	Dateinamenerweiterung vergessen, da z.B. bei eclipse automatisch erzeugt	Bed.2
14	Keine geeignete Methode bekannt	Y.1
15	Anzahl / Format der Übergabeparameter	Syn.6
16	Allgemeiner Ablauf unzureichend geklärt	C.1, X.2
17	Die Arbeit mit String und StringBuffer	Y.1
18	Einige Probleme bei der GUI-Programmierung	U
19	Fehlende Beispiele	D.1
20	Fehler beim Kompilieren des Programmes	E.3
21	Natürlich allgemein das Problem das Programm richtig aufzubauen und zu strukturieren	C.1
22	Problem eine richtige Verbindung zwischen den einzelnen Klassen herzustellen	D.3
23	Weiß nicht wie was bestimmtes geht und finde es meisten nicht in der api	Y.1
24	Aufteilung der Arbeit	C.2
25	Tutor versteht nicht oder kann nicht helfen	Y.2
26	Zeit reicht nicht um öffentliche Foren zu nutzen	Y.1, X.1
27	Suche nach fertigem Satzanfang länger als Tippen eines eigenen	Z, Bed.6
28	Hab den Stoff einfach nicht drauf	Y.1
29	Syntax, die nicht korrekt ist, weil man etwas vergessen hat	E.3
30	Unverständliche Compilerausgaben (z.B. Class expected mitten im Irgendwo)	E.3
31	Einbinden neuer unbekannter Programmelemente	D.1, B.2

Die Spalte *Problembeschreibung* enthält die Aussagen der Teilnehmer zu ihnen bekannten Problemsituationen, in der Spalte *Kategorie* sind diejenigen Problemkate-

[3] Außer Wertung, da TN sich auf Eclipse-IDE bezieht.

gorien notiert, denen die genannten Probleme zuzuordnen sind. Da die Befragung zeitlich entkoppelt vom Auftreten der jeweiligen Problemsituationen stattfand, war eine eindeutige Zuordnung in einigen wenigen Fällen nicht möglich: Das in Zeile 16 beschriebene Problem kann sowohl auf Schwierigkeiten bei der Definition von Teilaufgaben (C.1) hinweisen als auch durch Probleme bei deren Koordination (X.2) begründet sein.

Ferner wurde stets versucht, Probleme so konkret wie möglich einzuordnen. Wo dies nicht möglich war, wurden Oberkategorien genannt: In Zeile 18 äußert sich ein Teilnehmer, dass er *„einige Probleme bei der GUI-Programmierung"* hatte. Diese Beschreibung ist jeodch so allgemein gehalten, dass daraus zwar auf ein Problem in der Umsetzungsphase geschlossen werden kann, eine spezifischere Einordnung ist jedoch aufgrund fehlender detaillierterer Angaben nicht möglich.

Wenngleich auch die Zuordnung nicht immer eindeutig erfolgen konnte, so ließen sich doch sämtliche genannten Problemsituationen den vereinbarten Problemkategorien zuordnen. Die resultierende Verteilung der Problemnennungen auf die Problemkategorien ist in der Tabelle 11.2 zusammengefasst.

Tab. 11.2: Verteilung der Teilnehmerprobleme auf Problemkategorien

Problemkategorie	Anzahl
Arbeitsvorbereitung	-
B.Lernkontext	-
B.1	1
B.2	2
C.Problemlösungsorganisation	-
C.1	2
C.2	1
Umsetzung	2
D.Rohfassung	-
D.1	2
D.2	1
D.3	1
E.Fehlerbereinigung	1
E.3	5
E.8	1
Syntax	-
Syn.4	1
Syn.6	2
Bedien	-
Bed.2	2
Bed.6	1
F.Abschluss	-
F.2	1

Fortsetzung auf nächster Seite

Tab. 11.2: Verteilung der Teilnehmerprobleme auf Problemkategorien *Forts.*

Problemkategorie	Anzahl
Unterstützung	-
X.Koordination	-
X.1	1
X.2	1
Y.Informationsbeschaffung	-
Y.1	6
Y.2	1
Z.Fachkommunikation	1
	$\Sigma = 36$

Auffällig ist, dass sämtliche Nennungen sich ausnahmslos auf den fachlichen Prozess beziehen, wohingegen Problemsituationen des sozialen Prozesses gänzlich ungenannt bleiben. Von den insgesamt 36 Zuordnungen entfallen sechs (=16,67 %) auf die Arbeitsvorbereitung, zehn (=27,78 %) auf die Unterstützungsprozesse und 20 (=55,56 %) auf die Umsetzungsphase. Mehr als die Hälfte aller genannten Probleme sind somit der eigentlichen Programmiertätigkeit zuzuordnen. Innerhalb des Unterstützungsprozesses entfallen 70 % aller Nennungen auf die Informationsbeschaffung – ein deutlicher Indikator für die Relevanz dieser Aktivität hinsichtlich des Lernprozesses.

11.2 Identifikation von Problemsituationen

Das Konzept zur Erkennung von Problemsituationen während der synchronen, verteilten Zusammenarbeit beruht – wie auch die in Kapitel 10 behandelte Rollenanalyse – auf der Betrachtung der von den Teilnehmern während einer Sitzung ausgelösten Aktionen. Analog zur Rollenanalyse liegt auch bei der Problemanalyse der Schwerpunkt auf den CLS-Codes der ausgelösten Benutzeraktionen, jedoch werden bei Bedarf auch weitere verfügbare Informationen der Nachrichtenobjekte für die Untersuchung herangezogen.

11.2.1 Von Teilnehmern ausgelöste Probleme

Mit dem Ziel, diejenigen Problemsituationen, in denen ein Tutor zwecks Förderung des Arbeits- und Kernprozesses eingreifen sollte, aufzudecken und Erkenntnisse für eine zu konzeptionierende Problemerkennungskomponente zu gewinnen, wurden im Januar und Februar 2006 weitere acht Benutzertests[4] durchgeführt. An diesen nahmen 14 Studierende teil, die sich auf eine Zweier- und vier Dreiergruppen verteilten. Um möglichst viele Informationen über Problemsituationen zu erhalten, wurden

[4] Tatsächlich fanden im Rahmen dieser Studie neun Sitzungen statt, von denen jedoch eine (S_{70}) aufgrund technischer Probleme nicht in die Betrachtungen einfließen kann.

alle Teilnehmer instruiert, über die Kommunikationskomponente (s. Kap.8.2.4; s. Abb.8.12) den CLS-Code 34 (*„Lasst uns den Tutor fragen."*) auszulösen, wann immer sie der Meinung sind, in einer Problemsituation tutorielle Unterstützung zu benötigen.

Den Teilnehmern wurde suggeriert, dass sie dabei mit einem maschinellen System agieren, welches mittels der Benutzertests bezüglich seines Verhaltens auf Problemmeldungen evaluiert werden soll. Tatsächlich wurde die Arbeitsweise des scheinbar virtuellen Tutors durch einen menschlichen Operator simuliert. Um ein einheitliches Reaktionsverhalten des Operators sicherzustellen, wurde dieser entsprechend instruiert und bekam in speziellen Übungssitzungen zusätzliche Trainingsmöglichkeiten hinsichtlich seines Eingreifens in die Teamarbeit. Mit diesem *Wizard-of-Oz*-Experiment kann ein System, das in einer nur unvollständigen Umsetzung vorliegt, unter relativ realistischen Bedingungen getestet werden. Darüber hinaus ist es möglich, zusätzliche Anforderungen zu ermitteln, die in dem bisherigen Entwurf keine ausreichende Berücksichtigung finden. Der (menschliche) Tutor, der bei allen Sitzungen anwesend war und die Rolle des virtuellen Tutors simuliert hat, hat darüber hinaus – von den Teilnehmern unbemerkt – diejenigen Situationen protokolliert, in denen ein Tutor üblicherweise eingreifen würde. Insgesamt wurden während der acht im genannten Zeitraum durchgeführten Benutzertests von den Teilnehmern sechs Problemsituationen durch Auslösen von CLS-Code 34 an das System gemeldet (s. Tab.11.3).

Tab. 11.3: Von Teilnehmern ausgelöste Problemmeldungen

Meldung	Sitzung	Lfd.Nr.	Zeit	Teilnehmer
1	S_{69}	643	0:30:36	T_{305}
2	S_{69}	3924	1:16:44	T_{303}
3	S_{71}	940	0:39:31	T_{293}
4	S_{74}	1003	0:18:34	T_{286}
5	S_{76}	3091	1:11:57	T_{305}
6	S_{76}	3643	1:29:55	T_{305}

Für jede der sechs aus den Sitzungen S_{69} bis S_{77} resultierenden Problemmeldungen ist ersichtlich, in welcher Sitzung sie stattfand, zu welchem Zeitpunkt – relativ zum Sitzungsbeginn – sie ausgelöst wurde (Spalte *Zeit*) und welcher Teilnehmer ein Problem meldet. Ferner kann der Tabelle 11.3 entnommen werden, die wievielte Benutzeraktion innerhalb der jeweiligen Sitzung die Problemmeldung darstellt (Spalte *Lfd.Nr.*). Diese sechs Problemmeldungen werden durch neun weitere Problemsituationen ergänzt, die vom Tutor während der Sitzungen identifiziert werden konnten.

Es folgt zunächst eine eingehende Betrachtung aller sechs Problemmeldungen, wobei auch die jeweilige Vorgeschichte Berücksichtigung findet, um daraus potentielle Hinweise für die automatische Erkennung von Problemsituation durch das

VitaminL-System ableiten zu können. Anschließend werden in gleicher Weise die vom Tutor protokollierten neun Problemsituationen diskutiert.

11.2.1.1 Problemmeldung 1: Nicht erkannte Teilprobleme

Teilnehmer T_{305} fordert in Sitzung S_{69} zum Zeitpunkt $t_{643} = 0:30:36$ tutorielle Unterstützung an und spezifiziert sein Problem mit einem Folgebeitrag (zum Zeitpunkt $t_{802} = 0{:}31{:}26$) etwas genauer: *„Kannst Du mir mehr sagen bzw. mal ein beispiel geben zur verwendung von „instance of", bitte *augenaufschlag* ?"*. Bei nährerer Betrachtung der dieser Meldung vorausgehenden Benutzeraktionen des Teilnehmers T_{305} (s. Tab.E.2) sieht man, dass T_{305} das Problem bereits zum Zeitpunkt $t_{39} = 0{:}16{:}16$ formuliert hat, aber zunächst versucht hat, mit Hilfe der Vorlesungsskripte einen Lösungsansatz zu finden (t_{66}: *„Ich bin nicht sicher, lese gerade mal im skript nach. Wir hatten da neulich was mit Instanz erzeugen."*). Zwischenzeitlich beteiligte sich T_{305} immer wieder mit kommunikativen Beiträgen an der Gruppenarbeit. Unmittelbar vor Meldung der Problemsituation navigiert der Teilnehmer scheinbar ziellos zwischen diversen geöffneten Quelltextdokumenten hin und her (t_{528}ff).

Das Problem selbst lässt sich durch ein Missverstehen von Teilen der Aufgabenstellung begründen: Die Aufforderung *„Jedesmal, wenn Sie im Frame „Getraenkeautomat" ein Getränk auswählen, wird eine neue Instanz dieser Klasse erzeugt."* besagt, dass (unter Verwendung des `new`-Operators von Java) ein Objekt einer bestimmten Klasse (hier: `Getraenkeautomat`) generiert werden soll, der Teilnehmer T_{305} assoziiert den Passus *„eine Instanz [von] dieser Klasse"* fälschlicherweise mit dem Java-Schlüsselwort `instanceof`. Hierbei handelt es sich jedoch um einen (binären) Operator, mit dessen Hilfe überprüft werden kann, ob ein gegebenes Objekt von einem bestimmten Datentyp (genauer: von einer bestimmten Klasse oder Schnittstelle) ist – neue Objekte hingegen können damit nicht erzeugt werden. Offensichtlich herrscht folglich Unklarheit hinsichtlich einzelner Teile der Aufgabe, so dass diese Problemsituation der Problemkategorie A.3 (s. Kap.11.1.1.1) zuzuordnen ist.

Dass es zu diesem Problem kam, wundert nicht, wenn man das Verhalten aller Beteiligten etwas genauer betrachtet (s. Tab.E.3). Die Sitzung S_{69} beginnt mit einer zwölfminütigen Phase, welche die Teilnehmer zum individuellen Studium der Aufgabenstellung nutzen. Die eigentliche Gruppenarbeit wird zwölf Minuten nach Sitzungsbeginn ($t_1 = 0{:}12{:}00$) durch den Beitrag *„Zusammenfassend wer will was machen?"* von T_{304} (s. Tab.E.3, Lfd.Nr.=1) initiiert, woraufhin die Teilnehmer T_{305} und T_{304} neue Quelltextdokumente erzeugen (t_2 und t_4). Anschließend findet ein kurzer Dialog zwischen diesen beiden Teilnehmern statt (t_9 - t_{12}) mit dem Versuch einer Aufgabenverteilung, der damit endet, dass sich T_{304} eine Aufgabe auswählt (t_{12}), sofort mit der Codierung beginnt (t_{13}ff) und die Verantwortung für die verbleibende Aufgabenverteilung den anderen beiden Teilnehmern überlässt: *„Ich denke ich fang mal mit dem Automaten-Fenster an. Einer von euch kann ja versuchen die Ereignisabhörer einzubauen"*.

Zu diesem Zeitpunkt kann die Arbeitsvorbereitung jedoch in keiner Weise als zufriedenstellend abgeschlossen betrachtet werden, da nicht nur die Aufgabenverteilung eher einseitig statt einvernehmlich stattfand, sondern darüber hinaus auch noch Unklarheiten hinsichtlich der Aufgabenstellung und enthaltener Teilprobleme bestehen: So bittet Teilnehmer T_{305} zum Zeitpunkt $t_{39} = 0{:}16{:}16$ die übrigen Teilnehmer um Mithilfe bei der Klärung von Teilen der Aufgabenstellung (*„Zur Ausarbeitung : Hat das evtl mit instance of zu tun? (1. Satz von 2.) ."*), woraufhin T_{303} seine Unterstützung zusichert ($t_{67} = 0{:}18{:}38$: *„Ich sehe, was Du sagen willst, ich les grad auch noch nach."*), während T_{304} sich weiterhin ausschließlich seinen Quelltexten widmet und sich an der Gruppendiskussion nicht beteiligt. In der Folge bleibt das Problem ungelöst und führt letztlich dazu, dass T_{305} tutorielle Unterstützung anfordert ($t_{643} = 0{:}30{:}36$: *„Lasst uns den Tutor fragen!"*).

Im weiteren Verlauf der Sitzung erfolgt zwar seitens T_{304} eine korrekte Erläuterung ($t_{1765} = 0{:}48{:}46$: *„Ich denke das heisst einfach nur, das dann ein neues fenster aufgeht. dann könnt ihr ja beide getränke fenster machen. sind ja schließlich fünf."*), die aber nicht zur zufriedenstellenden Klärung des Problems beiträgt, so dass Teilnehmer T_{305} schließlich resigniert und sich anderen Aufgaben zuwendet ($t_{2039} = 0{:}52{:}33$: *„Deshalb höre ich jetzt auch auf zu suchen und werde konstruktiv. habt ihr bestimmte sachen zu deligieren vielleicht? . . . "*).

11.2.1.2 Problemmeldung 2: Verteilung der Aufgaben

In derselben Sitzung fordert T_{303} zum Zeitpunkt $t_{2039} = 0{:}52{:}33$ die Unterstützung des Tutors an. Dem vorausgegangen war eine Diskussion unter den Teilnehmern hinsichtlich der konkreten Umsetzung einer bestimmten Methode, die im Zusammenhang mit der Verarbeitung von Ereignissen benötigt wird: Es herrscht innerhalb der Gruppe Unstimmigkeit darüber, wie der Zugriff auf die Methode am besten zu bewerkstelligen ist. Die einzelnen Kommunikationsbeiträge bis zum Aufrufen des Tutors sind als Extrakt der Sitzung S_{69} in der Tabelle E.4 zusammengefasst.

Auch diese Problemsituation, die schließlich zur Anforderung tutorieller Unterstützung führt, lässt sich – wie die im vorigen Kapitel 11.2.1.1 untersuchte Problemmeldung – als Problem der Arbeitsvorbereitung festmachen: Offensichtlich bereitet es den Teilnehmern Schwierigkeiten, Konsens hinsichtlich einer adäquaten Aufgabenverteilung herzustellen, da mehrere Lösungsmöglichkeiten für das konkrete Teilproblem[5] in Frage kommen.

Auf mehrere Beobachtungen soll in diesem Zusammenhang hingewiesen werden:

1. Zunächst einmal ist festzustellen, dass sich diese zweite Problemsituation unmittelbar an die erste anschließt und quasi aus derem erfolglosem Abschluss (t_{2039}) folgt.

[5] Hier: Implementierung einer Methode zur Ereignisverarbeitung und Realisierung des Zugriffs auf selbige

2. Die daran anschließende Aufgabenverteilung folgt in wesentlichen Zügen dem bereits im vorigen Kapitel skizzierten Schema: T_{304} weist Teilaufgaben zu (s. u.a. t_{2105} und t_{2136}) und widmet sich dann wieder seinen Quelltexten.

3. Klarheit beziehungsweise Einigkeit über das weitere konkrete Vorgehen (hinsichtlich der Implementierung der genannten *Abbruch*-Methode) herrscht nicht. So bemängelt T_{303} wenige Minuten vor Auslösen der Problemmeldung (t_{3754} = 1:12:14): *„Kannst Du erläutern, warum/wie du das meinst??? ich brauch die abbruch methode ja in meinem frame und so viel zu schreiben is das ja mal nicht, meien können es auch komplizierter machen als es is ;) ".*

4. Innerhalb des Versuchs, Aufgaben zu verteilen, tritt erschwerend ein Kommunikationsproblem in Form einer zeitlichen Überschneidung der Beitragsproduktion (vgl. GÖLDNER, 2005, S.65) auf (s. t_{2135}, t_{2136} und t_{2137}). Als Folge davon muss T_{304} seine Frage wenig später erneut formulieren: *„Zur Ausarbeitung T_{305}.willst du mal den Abbruch Button belegen? ich weiß zwar nicht ob das was bringt den in einer extra-klasse zu plazieren, aber dann hast du wenigstens was zu tun."* (t_{2598} = 1:02:45).

Insgesamt hinterlässt die Gruppe der Sitzung S_{69} hinsichtlich ihrer Vorgehensweise einen unstrukturieren Eindruck. Insbesodere die nur mangelhaft durchgeführte Arbeitsvorbereitung kann als Ursache für die Problemsituationen, die im weiteren Verlauf der Sitzung von den Teilnehmern gemeldet wurden, ausgemacht werden.

11.2.1.3 Problemmeldung 3: Lokalisierung von Compile-Fehlern

In der Sitzung S_{71} wendet sich Teilnehmer T_{293} zum Zeitpunkt t_{940} = 0:39:31 an den Tutor und spezifiziert sein Problem kurz darauf etwas genauer: *„Weisst Du woran es liegt, dass der Compiler bei AutomatFrame.java einen Fehler anzeigt?"*[6]. Der Teilnehmer konnte demnach einen Quelltext übersetzen und hat auch erkannt, dass dabei Fehler aufgetreten sind (t_{885} = 0:37:03: *„...hatte .java vergessen, aber es werden immer noch fehler angezeigt?"*), jedoch bereitete ihm deren Lokalisierung offensichtliche Schwierigkeiten. Im Vorfeld sind zwar bereits diverse kleinere Probleme aufgetreten (s. Tab.E.6), diese konnten jedoch alle soweit behoben werden, dass erst die Fehlermeldungen des Compilers zur Anforderung tutorieller Unterstützung führten.

Die Vorgehensweise von Teilnehmer T_{293} bis zur Supportanforderung lässt sich wie folgt skizzieren:

1. Zunächst beteiligt sich T_{293} an einer kurzen Diskussion, die sich mit dem zukünftigen Aussehen des zu erstellenden Programms befasst (t_{12} = 0:02:27: *„Denkst Du awt oder swing?"* bzw. t_{15} = 0:03:43: *„Ich denke auch auf jeden fall grid layout."*).

[6] Die Aktionen des Teilnehmers T_{293} der Sitzung S_{71} bis zum Auslösen der Problemmeldung sind in der Tabelle E.6 zusammengefasst.

2. In weiteren Kommunikationsbeiträgen findet der Versuch einer Aufgabenverteilung statt (s. t_{23}, t_{38}, t_{171}).

3. Kurze Zeit später kann willkürliches Navigieren zwischen mehreren Dokumenten beobachtet werden (t_{182}, t_{192}ff, t_{198} etc.), gefolgt von der Umbenennung eines als geeignet erachteten Beispieldokuments, so dass Dateiname (und Klassenname) der Anforderung der Aufgabenstellung entsprechen (t_{206}). Hierbei wurde die Dateiendung `.java` vergessen. Dieser Fehler wird jedoch zu einem späteren Zeitpunkt erkannt und korrigiert (t_{836}).

4. Es folgt eine kurze Phase der Quelltextbearbeitung (von t_{207} = 0:19:44 bis t_{387} = 0:21:56) mit anschließenden Aufrufen des Compilers (t_{462}, t_{466}, t_{694}), woraufhin die vergessene Dateiendung schließlich ergänzt wird (t_{836} = 0:35:52).

Der dann folgende Compiler-Aufruf (t_{839}) liefert letztlich die Fehlermeldung *„AutomatFrame.java:12: invalid method declaration; return type required GridLayoutJFrame( )"*, denn Teilnehmer T_{293} hat zwar das Dokument umbenannt und auch den Namen der enthaltenen Klasse entsprechend angepasst, jedoch nicht den zur Klasse gehörenden Konstruktor. Dieser wird nun – aufgrund seines vom Klassennamen abweichenden Namens – nicht mehr als Konstruktor erkannt, sondern wie eine gewöhnliche Methode behandelt, bei der ein Rückgabetyp (oder `void`) zwingend erwartet wird. Das Resultat ist die aufgetretene Fehlermeldung. Im weiteren Verlauf gelingt es T_{293} tatsächlich, diesen Fehler zu lokalisieren (t_{1144} = 0:48:59: *„Ich denke ich weiß jetzt woran es lag."*) und zu beheben (t_{1402} = 0:52:58: *„Ich denke wir müssen auch den methodennamen anpassen, hab ich gemacht aber läuft immer noch nicht."*). Dass nach der Korrektur weitere Fehler bemerkt werden, ist für die Vorgehensweise nach dem Code-and-Fix-Zyklus nicht ungewöhnlich.

Im Vergleich zu anderen Gruppen und deren Sitzungen soll auf zwei Besonderheiten hingewiesen werden (s. Tab.E.5):

1. Die Kommunikation findet *im Team* unter Einbeziehung *sämtlicher* Teilnehmer statt.

2. Das direkte Ansprechen von Teammitgliedern beim Verfassen (und Absendern) von Kommunikationsbeiträgen stellt den Regelfall dar. Es gibt – abgesehen von der Anfangsphase dieser Sitzung (bis t_{33} = 0:07:10) – nur wenige Beiträge, die nicht explizit an ein anderes Teammitglied adressiert sind.

Dieser fast vorbildlich zu bezeichnenden Teamkommunikation steht eine nur unzureichende Spezifikation von Teilaufgaben in der Arbeitsvorbereitungsphase gegenüber (s. t_{21} und t_{23} in Tabelle E.5). Dies kommt auch etwas später wieder zum Tragen, wenn sich Teilnehmer T_{293} beispielsweise über eine nur ungenügende Aufgabenverteilung beschwert (t_{38} = 0:08:22: *„Zur Rechtfertigung ic hab doch jetzt aber nichts zu tun."*) oder aber auch – als Ursache davon – die Planung des Gesamtvorgehens (als Teil der Arbeitsvorbereitung) anmahnt (t_{171} = 0:14:59: *„Kannst Du mir mehr sagen was du jaetzt überhaupt machst., vierlleicht sollten wir erstmal*

einen groben plan über das programm erstellen, weil so fühle ich mich gerade sehr unnütz?").

Auch bei der Erstellung der Rohfassung legt die Gruppe eine typische Vorgehensweise an den Tag:

1. Es wird zunächst ein bereits vorhandenes, lauffähiges Programm (bevorzugt aus den zur Lehrveranstaltung gehörenden Unterrichtsmaterialien und Beispielen) gesucht, das als Vorlage für die zu bearbeitende Aufgabe geeignet scheint (s. t_{19}, t_{34} und t_{178}).

2. Das Programm wird soweit geändert, dass es den Anforderungen der Aufgabenstellung genügt (t_{195}).

Im vorliegenden Fall führte diese Vorgehensweise auch zu den tpischen Fragestellungen (t_{195}: *„jetzt müssen wir es noch umbenennen und das andere löschen, weißt du, wie?"*) und Problemsituationen.

11.2.1.4 Problemmeldung 4: Suche nach Template I

Der Teilnehmer T_{286} wandte sich in der Sitzung S_{74} zum Zeitpunkt $t_{1003} = 0{:}18{:}34$ an den Tutor. Dem vorausgegangen war eine Diskussion unter den Teilnehmern (s. Tab.E.7), die typische Elemente einer Aufgabenverteilung aufweist. Trotz der unorganisierten, eher spontanen Vorgehensweise, die sich durch eine Vermischung aus Arbeitsvorbereitung und Umsetzung auszeichnet[7], kommt die Anforderung tutorieller Unterstützung zu besagtem Zeitpunkt eher überraschend:

- Die zu erledigende Teilaufgabe konnte vergleichsweise präzise formuliert werden ($t_{582} = 0{:}13{:}50$: *„Ich denke wir brauchen eine statische MEthode die alles einliest, aus der Datei herraus und eine die alles aus dem Array in die Textdatei einliest."*).

- Ferner wird die Problemlösung kurz darauf vom selbem Teilnehmer präsentiert, denn zum Zeitpunkt t_{1004} – also im unmittelbaren Anschluss an die Tutoranfrage – verkündet T_{286} den übrigen Teammitgliedern *„Verzeihung public FileReader(String fileName)"*.

Die Gruppe war also durchaus in der Lage, das Problem aus eigener Kraft zu lösen. Dies bestätigt sich durch den weiteren Diskussionsverlauf, in welchem T_{286} schließlich ein Beispiel identifizieren konnte, das als Template für die Problemlösung herhalten kann: *„Ich denke das mit dem lesen Zeile für Zeile kopier ich mal so aus der Vorlesung, da es da genauso vorhanden is wie wir das brauchen ;-)"* ($t_{1369} = 0{:}27{:}08$).

[7] Die nicht in der Tabelle E.7 aufgeführten Aktionen der Sitzung S_{74} bestehen größtenteils aus Editor-, Navigations- und Dateioperationen.

Auffallend ist eine vergleichsweise ausgeprägte Arbeitsvorbereitung, in der auch zukünftige Implementierungsdetails erörtert werden (s. t_{206}, t_{207} und t_{209}). Dabei findet ansatzweise auch eine Aufgabenverteilung statt, die jedoch aufgrund ihrer Unzulänglichkeiten in der Folge zu Unklarheiten führt (s. t_{402} = 0:11:28: *„Ich denke T_{296} hat deinen Bubblesort gleich fertig :-P"* und t_{432} = 0:11:58: *„Verzeihung T_{294} auch"*). Aus der eigentlichen Umsetzung erfolgen immer wieder Rücksprünge in die Arbeitsvorbereitung, um beispielsweise Teilaufgaben zu definieren (s. u.a. t_{2647} und t_{2987} in Tab.E.8).

Insgesamt zeigt sich an dieser Sitzung, dass auch das Nicht-Eingreifen des Tutors unter Umständen eine geeignete Unterstützungsstrategie darstellen kann.

11.2.1.5 Problemmeldung 5: Suche nach Template II

Der erste Kommunikationsbeitrag der Sitzung S_{76} wird von Teilnehmer T_{304} ausgelöst und eröffnet damit knapp sechs Minuten nach Sitzungsbeginn die Aufgabenverteilung (s. Tab.E.9; t_4 = 0:50:50: *„Zusammenfassend also wer will was machen?"*), die bereits dreieinhalb Minuten später von allen Teilnehmern einvernehmlich und mit verteilten Aufgaben – vorläufig – abgeschlossen wird (s. t_9, t_{13} und t_{14}). Zu späteren Zeitpunkten muss diese Phase abermals aufgegriffen werden, um die auszuführenden Teilaufgaben zu konkretisieren (s. t_{2063}, t_{2180}, t_{2345} und t_{2507}).Im weiteren Verlauf der Sitzung wenden die Teilnehmer wieder eine bereits bekannte Strategie an, indem sie versuchen, geeignete Beispiele (in der Lehrmaterialien der Veranstaltung) zu finden, um diese dann den Anforderungen der Aufgabenstellung anzupassen (t_{904} = 0:25:49: *„Zur Ausarbeitung T_{304}, in den beispielen zu v12 is doch bestimmt was, was du reinkopieren kannst... hab mich da auch schon kräftig bedient."* und t_{920} =0:27:04: *„Verzeihung ich bin da auch schon am gucken bei Bibliotheksortieren, aber ich muss ja erst mal rausfinden was ich da ändern muss damit es bei der aufgabe passt."*)

Diese Anpassung[8] erweist sich in der Folge als nicht ganz unproblematisch (s. u.a. t_{1018} = 0:35:11: *„Weisst Du ich bin grad ziemlich ratlos. in Bibliotheksortieren wird einmal sortieren.sort aufgerufen, aber ich finde da keine methode sort in sortieren. wär vielleicht schon gut wenn ich wüsste wo dieser befehl herkommt?"* und t_{1195} = 0:43:19: *„Zur Rechtfertigung sort gibt es in sortieren nicht. ich versuche einfach bubble sort und entweder es funktioniert mit strings oder nicht."*). Ein möglicher Lösungsansatz wird kurze Zeit später von T_{30} gefunden (t_{1577}= 0:46:34: *„Zusammenfassend in v12.zip gibt es 2 mal sortieren, eines davon gehört zur bibliothek und enthält auch das sort."*) Auch der Gesamtablauf kann von den Teilnehmern korrekt skizziert werden (t_{2180} = 0:54:33: *„Zusammenfassend T_{303} muss wohl erst lesedatei aufrufen, dann den kram in ein array, dann sortieren und dann dateischreiben, oder?"* und t_{2228} = 0:57:45: *„Ich bin nicht sicher ob das so richtig ist, aber: ich mach nu ne methode, die die namen liest und ein stringarray namens arrayunsorted ausgibt. meine zweite methode schreibenDatei braucht das stringarray*

[8] Hier: Adaption eines Sortieralgorithmus für ganze Zahlen an die Sortierung von String-Objekten.

arraysorted und schreibt in die textdatei.").

Als problematisch erweist sich hingegen eine andere Teilaufgabe, in welcher es um das zeilenweise Auslesen einer Textdatei und das Ablegen der eingelesenen Zeilen in ein String-Array geht: Diese Aufgabe führt letztlich zur Supportanfrage (t_{3091} = 1:11:57) von T_{305}. Kurz darauf (t_{3107} = 1:13:13) spezifiziert der Teilnehmer sein Problem etwas genauer: *„Bitte zeige mir ob es möglich ist, die zeilenanzahl in einer datei einfach so zu bestimmen!"* Wenig später kann T_{305} – ohne Hilfeleistung seitens der Teammitglieder oder eines Tutors – eine Lösung des Problems präsentieren, indem eine willkürliche konstante Obergrenze für die Array-Größe a priori festgesetzt wird: *„Zur Ausarbeitung ok, also ab jetzt die namenzahl nicht mehr ändern. das array hat halt ne länge von 10 und feddich"* (t_{3191} = 1:15:25). Interessanterweise hat T_{305} unmittelbar auf der Supportanforderung einen gleichermaßen korrekten wie flexiblen Lösungsansatz benannt, diesen jedoch aufgrund von Unsicherheiten bezüglich der Lerninhalte als Frage formuliert: *„Verzeihung kann es sein, dass ich zwei mal lesen muss, weil ich das string array ja erst erzeugen muss, bevor ich was reinschreib, aber zur erzeugung brauche ich ja die anzahl der elemente, also die zeilenzahl..."* (t_{3090} = 1:11:50).

Auch in diesm Fall hat das Nicht-Eingreifen des Tutors nicht geschadet, denn es konnte zu dem aufgetretenen Probleme letztlich eine praktikable Lösung gefunden werden – wenngleich diese Lösung nicht als bestmögliche Lösung angesehen werden kann.

Im Gegensatz zur Problemmeldung 1 (s. Kap.11.2.1.1), die ebenfalls vom Teilnehmer T_{305} ausgelöst wurde, ist dessen Verhalten im vorliegenden Fall gänzlich anderer Natur: Anstatt scheinbar orientierungslos zwischen diversen Dokumenten hin- und herzuwechseln (s. Tab.E.2: t_{1753}ff) legt der Teilnehmer nunmehr ein gänzlich unauffälliges Verhalten an den Tag und widmet sich aktiv seinem Quelltext, erkenntlich an einer Vielzahl von Editor- und weiteren Operationen, die typisch für das Bearbeiten von Quelltexten sind und in der halben Stunde vor der Problemmeldung von Teilnehmer T_{305} durch das System registriert werden konnten. Konkret bedeutet dies: Zwischen t_{1255} (0:44:11) und t_{3090} (1:11:50) hat T_{305} über 1200 Operationen ausgelöst, davon 672 Einfüge-Operationen (Code 37), 229 Markierungen (Code 52), 197 Lösch-Operationen (Code 38) und 107 Zeilenwechsel (Code 36). Ein Auszug diesen Zeitraum betreffend findet sich in der Tabelle E.10.

11.2.1.6 Problemmeldung 6: Fehlende Dateiendung

Die letzte Problemmeldung dieser Versuchreihe wurde ebenfalls in der Sitzung S_{76} von Teilnehmer T_{305} ausgelöst: Zum Zeitpunkt t_{3643} (1:29:55) erfolgt die zweite Anforderung tutorieller Unterstützung durch T_{305} in Sitzung S_{76}, gefolgt von zwei weiteren Beiträgen zur näheren Spezifizierung des aufgetretenen Problems (t_{3645} = 1:30:27: *„Kannst Du erläutern, warum/wie ich irgendwie nich rauskriege, wo die fehler in meinem quelltext sind?"* und t_{3657} = 1:31:44: *„Kannst Du erläutern, warum/wie ich die fehler beim compilieren sehen kann?"* ; s. Tab.E.12).

Die Vorgeschichte dieser Fehlermeldung schließt sich unmittelbar an das Ende der vorigen Fehlermeldung an (s. Kap.11.2.1.5): Zum Zeitpunkt t_{3191} (1:15:25) legt sich T_{305} auf eine Lösung fest und tritt dann in eine Umsetzungsphase gemäß Code-and-Fix-Zylus ein[9], die nach einem Aufruf des Compilers (t_{3542} = 1:23:20) zu einer Problemsituation führt (t_{3551} = 1:24:52: *„Zusammenfassend e fehler vom compiler sehe ich nicht. wat mach ich schussel denn nu wieder falsch?"*). In den nachfolgenden fünf Minuten bis zur Meldung dieses Fehlers an den Tutor (t_{3643}) erfolgt ein weiterer Aufruf des Compilers (t_{3613}), begleitet von mehreren Dokumentenwechseln (Code 35 zu t_{3552}, t_{3553}, t_{3612}, t_{3614} und t_{3615}).

Eine Erläuterung zum aufgetretenen Fehler erfolgt knapp zwei Minuten nach deren Meldung seitens Teilnehmer T_{303}, der sich in einer vergangenen Sitzung bereits mit demselben Problem konfrontiert sah: *„Ich sehe, was Du sagen willst T_{305} ich hatte das problem letzte woche auch als ich vergessen hatte beim speichern .java hinter die datei zu setzen..."* (t_{3656} = 1:31:32; s. Tab.E.11). Mit diesem Beitrag konnte T_{305} in der Folge den Fehler[10] durch Anfügen der vergessenen Dateiendung `.java` korrigieren (t_{3796} = 1:38:46) und in zur Phase der Bereinigung der vom Compiler angezeigten Fehlermeldungen übergehen. Ein Einschreiten eines Tutors war somit auch in dieser Problemsitutation nicht notwendig, der Fehler konnte innerhalb der Gruppe gelöst werden.

11.2.2 Ergänzende Problembeobachtungen

Zusätzlich zu den von den Teilnehmern ausgelösten Problemmeldungen konnten aus den Sitzungen weitere Problemsituationen extrahiert werden[11]. Dazu wurden die zugehörigen Sitzungsprotokolle (s. Kap.8.4.4) mittels intellektueller Analyse und unter Zuhilfenahme des Logfile-Analyzers (s. Kap.8.4.4.3.1) auf Probleme hin begutachtet, die einer erfolgreichen Fortführung der Aufgabenbearbeitung im Weg standen, aber von den Teilnehmern entweder nicht als solche erkannt oder zumindest nicht an den Tutor gemeldet wurden.

11.2.2.1 Beobachtung 7: Handhabung von Arrays

Während der Sitzung S_{71}, in deren Verlauf bereits ein Teilnehmer eine Problemmeldung bezüglich der Lokalisierung von Compile-Fehlern ausgelöst hatte (s. Kap.11.2.1.3), traten darüber hinaus Probleme in der Handhabung von Arrays auf: Konkret wurden laut Aufgabenstellung mehrere gleichartige `Button`-Objekte benötigt, die sehr ähnliches Verhalten besitzen und sich lediglich hinsichtlich ihrer Beschriftung unterscheiden sollten. Im Verlauf der Sitzung wurde das ursprüngliche

[9] Diese Phase dauert ca. acht Minuten und beinhaltet 314 Operationen, davon 91 Markierungen (Code 52), 71 Einfüge-Operationen (Code 37) und 64 Zeilenwechsel (Code 36).

[10] Der aufgetretene Fehler ist der Problemkategorie Ü.6 (Klassen und Methoden; s. Kap.11.1.1.2.2) zuzurechnen und wurde durch eine Verletzung der Konvention der Übereinstimmung von Klassenname und Dateiname ausgelöst.

[11] Um etwaigen Missverständnissen vorzubeugen, wurde die in Kapitel 11.2.1 begonnene Numerierung der Probleme fortgesetzt.

Beispiel aus dem Internet (s. Kap.11.2.1.4; s. t_{19} in Tab.E.5) modifiziert, so dass zum Zeitpunkt t_{1402} folgendes Code-Fragment vorlag:

Beispiel 11.1: Problembehafteter Quellcode (Problem 7)

```
...
// create buttons and add them to this container
buttons = new JButton[6];
for( int i = 0 ; i < buttons.length ; i++ )
{
  buttons[i]
   = new JButton( "Cola ""Fanta ""Sprite ""Sekt ""Bier ""Abbruch "+(i+1) );
  getContentPane().add( buttons[i] );
}
...
```

■

Der Übersetzungsversuch mit dem Compiler durch Teilnehmer T_{292} zum Zeitpunkt t_{1421} (0:53:40) liefert das nachfolgend dargestellte Resultat (s. Beispiel 11.2).

Beispiel 11.2: Übersetzen fehlerhaften Quellcodes (Problem 7)

```
User T292 hat das Dokument 'AutomatFrame.java' compiliert
 -  dabei sind Fehler aufgetreten!
AutomatFrame.java:36: ')' expected
      buttons[i]
       = new JButton( "Cola ""Fanta ""Sprite ""Sekt ""Bier ""Abbruch "+(i+1) );
                                ^
1 error
```

■

Es schließt sich die aus der Sitzung extrahierte Diskussion – vornehmlich zwischen T_{292} und T_{293} – an, in welcher diverse Elemente des betroffenen Code-Fragments erörtert werden (s. Tab.E.13). Dabei zeigt sich auch, dass der von den Teilnehmern gewählte Ansatz, gemäß welchem bereits vorhandene und als geeignet erscheinende Beispiele als Grundlage für die Aufgabenbearbeitung gewählt werden, durchaus Schwachpunkte besitzt: Einzelne Aussagen der Teilnehmer lassen darauf schließen, dass auch die Verwendung von Beispielen ein Mindestmaß an Grundverständnis der enthaltenen oder anzuwendenden Konzepte und Sprachkonstrukte erfordert (s. t_{2540}: *„Lasst es mich so erklären das stand schon so in dem programm, aber ich denke, dass ja immer ein vbutton dazukommen muss."* oder auch t_{2553}: *„Zusammenfassend naja, die schleife war doch aber schon vorher da."*). Dementsprechend gestaltet sich auch der aus dieser Diskussion resultierende Korrekturversuch, der nachfolgend dargestellt ist (s. Beispiel 11.3).

Beispiel 11.3: Korrigierter Quellcode (Problem 7)

```
...
// create buttons and add them to this container
buttons = new JButton[5];
for( int i = 0 ; i < buttons.length ; i++ )
{
  buttons[i] = new JButton( "Cola "+"Fanta "+"Sprite "
```

```
                              +"Sekt "+"Bier "+"Abbruch "+(i+1) );
  getContentPane().add( buttons[i] );
}
...
```

■

Mit dieser Version haben die Teilnehmer zwar eine syntaktische Korrektur des ursprünglich fehlerbehafteten Quelltexts erzielt, ein konzeptionelles Problem der richtigen Handhabung von Arrays und enthaltenen (`JButton`-)Elementen bleibt jedoch weiterhin bestehen. Auch der weitere Verlauf der diesem Problem zugeordneten Diskussion (s. Tab.E.13) führt zu keiner brauchbaren Lösung, wie sie beispielsweise in dem folgenden Lösungsvorschlag dargestellt ist (s. Beispiel 11.4).

Beispiel 11.4: Lösungsvorschlag (Problem 7)

```
...
// define array with button-names
final String [] BUTTONNAMES
 = { "Cola", "Fanta", "Sprite", "Sekt", "Bier", "Abbruch" };
// create buttons and add them to this container
buttons = new JButton[BUTTONNAMES.length];
for( int i = 0 ; i < buttons.length ; i++ )
{
  buttons[i] = new JButton( BUTTONNAMES[i] );
  getContentPane().add( buttons[i] );
}
...
```

■

11.2.2.2 Beobachtung 8: Bedienung des Interpreters

Noch während die zuvorgenannte Problemsituation 7 (s. Kap.11.2.2.1) präsent ist und die Teilnehmer sich um deren Bereinigung bemühen:, tritt innerhalb der Sitzung S_{71} interessanterweise ein weiteres Problem auf: Teilnehmer T_{292} zeigt leichte Probleme bei der Bedienung der VitaminL-IDE, insbesondere beim Aufrufen des Interpreters. Explizit teilt der Teilnehmer dies dem restlichen Team zum Zeitpunkt t_{1631} (0:59:22) mit: *„Weisst Du was ich bei klasse eingeben muss?"* (s. Tab.E.14), die eigentliche Ursache liegt aber im Aufruf des integrierten Interpreters zum Zeitpunkt t_{1143} (0:48:56).

In der sich anschließenden Diskussion gelingt es dem Team, unter zeitweiliger Mitwirkung eines anwesenden Tutors (Teilnehmer T_{284}), dieses Problem zur Zufriedenheit aller Beteiligten zu klären (s. t_{2055}, t_{2066} und t_{2076}). Das Eingreifen des Tutors in dieser Situation hatte nur unterstützenden Charakter und die Dauer der Problembearbeitung etwas verkürzt, denn es gibt Belege dafür, dass das Team auch ohne Tutor das Problem zufriedenstellend hätte lösen können, denn bereits nach kurzer Diskussion zwischen T_{292} und T_{293} liefert T_{293} die Lösung: *„Lasst es mich so erklären da muss doch der name unserer klasse rein!!!"* (t_{1787} = 1:01:53). Im weiteren Verlauf werden Details geklärt und Alternativen zum integrierten Interpreter angeboten (Ausführung in der Shell; s. t_{2051}). Nach erfolgreichem Abschluß dieser Problemsituation (t_{2076}) wird das bis dato noch offene, ungeklärte Problem 7 wieder aufgegriffen (s. Tab.E.13).

11.2.2.3 Beobachtung 9: Bedienung des Compilers

Im Laufe der Sitzung S_{73} wendet sich Teilnehmer T_{312} zum Zeitpunkt t_{3958} (0:38:18) mit einem Bedienproblem der VitaminL-IDE (Kategorie N.4; s. Kap.11.1.1.2.2) an Teilnehmer T_{311}: *„Verzeihung Hast Du Das Programm schon mal laufen lassen, weiß gerade nicht den Ausführungsbefehl"* (s. Tab.E.15), woraufhin T_{312} zunächst als Alternative zur VitaminL-IDE auf die Nutzung der Shell verweist (t_{3897}), um dann wenig später feststellen zu müssen, dass sämtliche offenen Dokumente ohne die notwendige Dateiendung `.java` gespeichert wurden. Nach Erkennen dieses Fehlers wurden benötigte Dokumente umbenannt und dabei um die fehlende Endung `.java` ergänzt.

Dass es ein Problem gibt, hätte unter Umständen schon viel früher auffallen müssen: Ab dem Zeitpunkt t_{1422} (0:13:03) ruft T_{312} innerhalb von 52 Sekunden sechsmal den Compiler auf (s. Tab.E.16), was jedoch aufgrund der fehlenden Dateiendung zu keinem sichtbaren Erfolg führt[12]. Erst 24 Minuten später (t_{3889}) wendet sich T_{312} an sein Teammitglied T_311, die Zwischenzeit wurde mit dem Editieren von Dokumenten überbrückt (s. Tab.E.16). Die Wartezeit auf eine Antwort (t_{3890}ff) ist geprägt von einer Folge von Navigationsoperationen (CLS-Codes 35 und 36) sowie einem weiteren (erfolglosen) Compile-Versuch (t_{3895}) mit anschließendem Aufruf des Interpreters (t_{3896}). Ob das Lösungsangebot, auf die Shell auszuweichen, auch tatsächlich in Anspruch genommen wurde, ist anhand der Protokolle nicht ersichtlich, jedoch lässt die zeitliche Abfolge der Benutzeraktionen von T_{312} darauf schließen, dass dies nicht geschah. Vielmehr folgen weitere Navigationsoperationen (t_{3912}, t_{3917}f und t_{3939}ff), bevor sich T_{312} letztlich mit seinem Problem direkt an den Tutor wendet: *„Weisst Du welche Parameter ich zur Programmeingabe beim Ausführen eingeben muss?"* (t_{3992}; s. Tab.E.15).

Die Problemsituation äußert sich zwar als Bedienproblem von Compiler und Interpreter, es drängt sich jedoch die Vermutung auf, dass seitens T_{312} eine Verwechslung von Compiler und Interpreter vorliegt oder aber schlichtweg eine unscharfe respektive unpräzise Formulierung des Problems.

11.2.2.4 Beobachtung 10: Lokalisieren von Compile-Fehler

Im weiteren Verlauf der Sitzung S_{73} werden Compile-Fehler bemerkt und im Team diskutiert, jedoch kann keine Lösung erzielt werden, so dass sich Teilnehmer T_{311} schließlich an den Tutor wendet: *„Kannst Du erläutern, warum/wie Zeile 63 (e.getSource) nicht in Ordnung ist?"* (s. Tab.E.17). Der Fehler, der auftritt, als Teilnehmer T_{311} zum Zeitpunkt t_{4002} den Compiler aufruft (s. Tab.E.18), ist nachfolgend wiedergegeben (s. Beispiel 11.5).

[12] In einer Shell führt der Aufruf des Compilers mit einer Datei ohne Endung zu einem Fehler, in dessen Folge eine Übersicht aller verfügbaren Compile-Optionen ausgegeben wird. Konzeptbedingt leistet die VitaminL-IDE dies zum Zeitpunkt der Untersuchungen nicht, sondern unterdrückt jene Ausgaben, so dass Benutzer nicht explizit auf einen Fehler hingewiesen werden.

Beispiel 11.5: Übersetzen fehlerhaften Quellcodes (Problem 10)

```
User T311 hat das Dokument 'AutomatFrame.java' compiliert
 -  dabei sind Fehler aufgetreten!
AutomatFrame.java:64: cannot find symbol
symbol  : variable button
location: class AutomatFrame
     if( e.getSource() == button[5] )
                          ^
1 error
```

Diesem Fehler liegt (in Auszügen) folgender Quelltext zugrunde:

Beispiel 11.6: Problembehafteter Quellcode (Problem 10)

```
...
public class AutomatFrame extends JFrame implements
    WindowListener,ActionListener
{
  public AutomatFrame()
  {
    super();
    ...
    String buttonNames[]
     = {"Cola", "Fanta", "Sprite", "Sekt", "Bier","Abbruch"};
    JButton buttons[] = new JButton[6];
    for (int i=0; i<buttons.length; i++)
    {
      buttons[i] = new JButton(buttonNames[i]);
    }
    ...
  }

  /**
   * Invoked when an action occurs
   */
   public void actionPerformed(ActionEvent e)
   {
     // Stammt das Ereignis vom Button btnEnde?
     if( e.getSource() == buttons[5] )
     {
       ende();
     }

   }

  ...
}
```

■

Wie der in Tabelle E.17 zusammengefassten Diskussion entnommen werden kann, werden (falsche) Mutmaßungen hinsichtlich der Verwendung von Arrays geäußert (t_{5146}: *„Ich sehe, was Du sagen willst ich glaube da steckt keine Angabe vom Array [ ] mit im Button-Befehl"*). Tatsächlich aber handelt es sich um ein Problem hinsichtlich des Gültigkeitsbereiches von Variablen (Kategorie Ü.4; s.

Kap.11.1.1.2.2): Die im Konstruktor der Klasse `AutomatFrame` definierten Variablen (`JButton buttons[]`) haben nur lokale Gültigkeit, trotzdem erfolgt in der Methode `actionPerformed(...)` ein Zugriffsversuch. Abhilfe würde die Definition von `JButton buttons[]` als Attribut bringen.

Neben der ergebnislosen Diskussion der Teilnehmer, die letztlich zum Anrufen des Tutors führt, können auch hier auffällig viele Navigationsoperationen sowohl nach dem Aufruf des Compilers (t_{4002}) in Tabelle E.18) als auch vor und nach der Anforderung der tutoriellen Unterstützung (t_{5674}) beobachtet werden.

11.2.2.5 Beobachtung 11: Anpassung eines Templates

In der Sitzung S_{74} findet, beginnend zum Zeitpunkt t_{1577}, eine Diskussion zwischen den Teilnehmern T_{286}, T_{294} und T_{296} statt, die sich mit der Anpassung eines Templates befasst (s. Tab.E.19). Konkret geht es bei dieser Template-Anpassung um einen in der Vorlesung vorgestellten Algorithmus zum Sortieren ganzer Zahlen, der im Rahmen der gestellten Aufgabe nunmehr zur Sortierung von Zeichenketten Verwendung finden soll; es ist folglich eine Adaption des Verfahrens vom Datentyp `int` zum Datentyp `String` erforderlich. Nach der Diskussion die sich – unterbrochen von Recherche und Dokumentenbearbeitung – über einen Zeitraum von knapp einer Stunde erstreckt, einigen sich die Teilnehmer auf eine vermeintliche Lösung, welche auf dem Vergleich der Anfangsbuchstaben der zu sortierenden `String`-Objekte (`String.charAt(0)`; s. t_{5534}ff in Tab.E.19) basiert.

Es muss somit festgestellt werden, dass die Teilnehmer trotz richtiger Ansätze wie dem Hinweis auf die Schnittstelle `Comparable` (t_{2081}: *„Aber haste schon mal unter Compareable gesucht?:"*)gelangen die Teilnehmer letztlich nur zu einer teilweise funktionierenden Lösung[13]. Unterstützung durch einen Tutor, der auf diesen Umstand hinweist, wäre somit in dieser Situation durchaus angebracht.

Um Hinweise für die (automatische) Identifikation dieser Problemsituation – und damit Anhaltspunkte für ein mögliches Eingreifen eines Tutors – zu erhalten, ist es hilfreich, die Aktionen einzelner Teilnehmer im Kontext dieser Problemsituation genauer zu betrachten: Bei Teilnehmer T_{294} können beispielsweise Anhäufungen von Navigationsoperationen im (zeitlichen) Umfeld seiner Kommunikationsbeiträge zu Beginn der Problemsituation (t_{1577} und t_{2044}) festgestellt werden (s. Tab.E.20). Dies deckt sich durchaus mit anderen im Rahmen dieser Studie diskutierten Problemsituationen.

11.2.2.6 Beobachtung 12: Suche nach Template III

Bei Betrachtung des in Tabelle E.21 wiedergegebenen Ausschnitts aus der Kommunikation von Sitzung S_{75} drängt sich die Schlussfolgerung auf, dass der Teilnehmer T_{292} entweder Probleme damit hat, den Lernkontext herzustellen respektive einen

[13] Der Vergleich von Anfangsbuchstaben funktioniert, solange keine zwei `String`-Objekte mit demselben Anfangsbuchstaben zu sortieren sind.

Zusammenhang zwischen der Aufgabenstellung und den Lehrmaterialien herzustellen, oder aber sich mit wie auch immer gearteten Schwierigkeiten beim Finden eines geeigneten Templates konfrontiert sieht. Statt dem Hinweis seines Teammitglieds T_{293} zu folgen (t_{829}: *„:es müsste doch aber in den sachen zur letzten vorlesung ein beispiel geben, oder nicht?"*), wird nach vergleichsweise kurzer (Bedenk-?)Zeit der Tutor konsultiert (t_{994}: *„Weisst Du wie der sortieralgorithmus fiunktioniert, oder kannst du mir ein beispiel geben?"*).

Auch in diesem Fall können bei näherer Betrachtung der Aktionen von Benuzter T_{292} Anhäufungen von Navigationsoperationen zwischen dem Hilfegesuch an das Teammitglied (t_{786}) und dem Anrufen des Tutors (t_{994}) festgestellt werden (s. Tab.E.22).

Interessanterweise wendet sich auch Teilnehmer T_{293} in dieser Sitzung lieber direkt an den Tutor, anstatt aufgetretene Probleme der Aufgabenbearbeitung im Team zu diskutieren oder mittels eigenständiger Recherche selbst zu erarbeiten. Dieses Vorgehen kann bei beiden Teilnehmern im Laufe der Sitzung mehrfach beobachtet werden (t_{999}, t_{1856}, t_{2442} und t_{2954}) – eine Antwort des Tutors bleibt in dieser Sitzung konzeptbedingt aus, dementsprechend moniert T_{293} auch *„Ich denke der Tutor ignoriert mich!! Hab schon 2mal was gefragt aber keine keine reaktion von ihm..."* (t_{2263}). Unter Umständen stellt für diesen Teilnehmer ein virtueller Tutor, der nur auf Anfrage reagiert, die bessere Unterstützungsstrategie dar als ein Tutor, der versucht, Problemsituationen zu detektieren, um dann selbständig – in Form entsprechender Hilfsangebote – in das Geschehen einzugreifen.

11.2.2.7 Beobachtung 13: Problem der Arbeitsvorbereitung

Die erste Problemsituation der Sitzung S_{77} konnte zum Zeitpunkt t_{253} (0:14:30) festgestellt werden: T_{288} wendet sich an sein Team mit der Bitte um Unterstützung bei der weiteren Bearbeitung seines bisher erzielten Arbeitsergebnisses (*„Zusammenfassend habe ich mal einen anfang gemacht und weiß jetzt aber nicht mehr weiter"*; s. Tab.E.23), das nachfolgend in Form der Quelltext-Datei `ReadAndWrite.java` wiedergegeben ist (s. Beispiel 11.7).

Beispiel 11.7: Quellcode des ersten Lösungsansatzes (Problem 13)

```
import java.io.*;

public class ReadAndWrite
{
  FileReade f;
  int c;

  try
  {
    f = new FileReade("....");
    while (( c= f.read()) !=-1)
    {
```

```
    }
  }
}
```
■

Dieses Code-Fragment enthält für die Aufgabenbearbeitung durchaus brauchbare Ansätze, ist jedoch aus der Sicht des Java-Compilers mit einigen Syntax-Fehlern behaftet. Kurze Zeit später präsentiert der Teilnehmer T_{288} seinen Teammitgliedern einen weiteren, ähnlichen Ansatz, den er als Beispiel (`Listing1809.java`) in der zur Lehrveranstaltung empfohlenen Literatur gefunden hat (vgl. KRÜGER, 2006, Kap.18.3). Er versucht anschließend, diesen an die Aufgabenstellung anzupassen (t_{261} und t_{263}). Das Ergebnis zum Zeitpunkt t_{572} ist in folgendem Quelltext wiedergegeben.

Beispiel 11.8: Überarbeiteter Quellcode (Problem 13)

```
import java.io.*;

public class ReadAndWrite
{
  LineNumberReader f;
  String line;

  try
  {
    f = new LineNumberReader(new FileReader(""));
    while ((line = f.readLine()) != null)
    {
      System.out.print(f.getLineNumber() + ": ");
      System.out.println(line);
    }
    f.close();
  }
  catch (IOException e)
  {
    System.out.println("Fehler beim Lesen der Datei");
  }
}
```
■

Obwohl auch dieser Quelltext noch einige Syntax-Fehler aufweist, ist T_{288} damit einer endgültigen Lösung wieder ein Stück näher gekommen. Dennoch fordert er wiederholt die Mithilfe seines Teams ein (t_{624}: *„Zusammenfassend wie soll es denn jetzt weitergehen?"*), woraufhin T_{290} nur eine Minute später das nächste Bauteil (s. Beispiel 11.9) zur Gesamtlösung beiträgt: *„Lasst mich Euch zeigen in Zeile drei steht die Methode die ihr zum Sortieren braucht"* (t_{626}).

Beispiel 11.9: Quellcode des zweiten Lösungsansatzes (Problem 13)

```
class Sort
{
  public String [] sortieren(String [] str)
  {
  }
}
```
■

Auf diese recht spezielle Weise, indem dem Team als geeignet erscheinende Templates vorgestellt werden, erarbeitet sich die Gruppe sukzessive die für die Lösung der gestellten Aufgabe notwendigen Konzepte sowie deren Umsetzung, ohne dass ein Eingreifen eines Tutors notwendig erscheint. Bei genauerer Betrachtung der Gruppensitzung, respektive der Kommunikationsbeiträge (s. Tab.E.23), ist deutlich erkennbar, dass die hier geschilderte Problemsituation ihren eigentlichen Ursprung im Beginn der Sitzung hat, der eine abgeschlossene Arbeitsvorbereitung vermissen lässt. Insbesondere findet eine Zerlegung des Problems in Teilaufgaben mit anschließender Aufgabenverteilung nicht in dem zu erwartenden Umfang statt. Dies monieren dann auch 2 der 3 Teilnehmer noch in der Anfangsphase der Sitzung (t_4: *„Ich denke wir sollten jetzt erstmal die aufgaben verteilen."* (T_{288}) und t_{26}: *„Ich denke wir müssen doch die aufgaben verteilen."* (T_{316})) – eine Korrektur des Vorgehens bewirken die Teilnehmer mit ihren Einwänden hingegen nicht: T_{290} stimmt diesem Vorschlag zwar zu (t_{31}: *„Ja."*), leitet aber die Sitzung relativ schnell in die Umsetzungsphase über (t_{10}: *„Ich denke Dann sollte sich* T_{316} *an den Strings versuchen."* und t_{172}: *„Deshalb* T_{316} *such dir einfach was aus, ich mach dann was übrig bleibt."*).

Mit dem Ziel, Hinweise für eine automatische Problemidentifikation zu erhalten, werden die Aktionen des betroffenen Teilnehmers (T_{288}) aus dem zeitlichen Umfeld der Problemmeldung (t_{253}) extrahiert und in Tabelle E.24 kompakt dargestellt. Entgegen bisherigen Beobachtungen kann dort (t_{241} - t_{255}) keine Anhäufung von Navigationsoperationen festgestellt werden. Dies ist nun nicht weiter verwunderlich, zeigt sich doch T_{288} nur wenig später in der Lage, mit dem in Beispiel 11.8 aufgeführten Quellcode eine erste Lösung für sein eigenes Problem zu präsentieren. Etwas anders sieht die Situation zum Zeitpunkt t_{624} aus, als T_{288} für die Bearbeitung derselben Teilaufgabe (Hier: Zeilenweises Einlesen aus einer Datei) erneut Unterstützung vom Team anfordert (*„Zusammenfassend wie soll es denn jetzt weitergehen?"*): Im Zeitraum von t_{608} (0:21:11) bis t_{666} (0:30:56) waren 18 der 22 aufgezeichneten Aktionen von Benutzer T_{288} Dokumentenwechsel, die restlichen vier Aktionen waren Kommunikationsbeiträge.

11.2.2.8 Beobachtung 14: Suche nach Template IV

Bereits während der Bearbeitung des vorigen Problems kann in der Sitzung S_{77} ein weiteres Problem identifiziert werden, das zum Zeitpunkt t_{610} (0:21:15) von Teilnehmer T_{316} eingebracht wird: *„Bitte zeige mir wo liegt die datei namen.txt!"* (s. Tab.E.25). Auffällig auch in dieser Situation ist die Häufung von Navigationsoperationen des Benutzers T_{316}, der im Zeitraum von t_{589} (0:20:44) bis t_{640} (0:24:30) neben dem bereits zitierten Kommunikationsbeitrag acht Dokumentenwechsel tätigt, sonst aber keine weiteren Aktionen auslöst.

Als Antwort auf das Hilfegesuch von T_{316} sagt Teammitglied T_{290} seine Unterstützung zu: *„Ich denke Ich schau mal wie sich das mit der Pfadangabe machen"* (t_{645}). In der Folgezeit – über einen Zeitraum von etwa fünfeinhalb Minuten – werden keine Aktivitäten von T_{290} durch das System registriert; es darf daher ange-

nommen werden, dass während dieses Zeitraums recherchiert wird[14]. Im Anschluss an diese Phase (t_{667}) widmet sich der Teilnehmer T_{290} zunächst der Bearbeitung des benötigten Sortieralgorithmus, um sich nach weiteren 18 Minuten (t_{1720}) schließlich an den Tutor zu wenden (*„Kannst Du erläutern, warum/wie ich eine relative Pfadangabe schreibe?"*). Der Anruf des Tutors erfolgt hierbei – im Unterschied zu anderen Beobachtungen – unmittelbar auf die Bearbeitung einer Teilaufgabe (hier: Sortieralgorithmus) und ist nicht von Navigationsoperationen begleitet; diese werden in sehr geringem Maße erst nach der Anforderung tutorieller Unterstützung durchgeführt (s. Tab.E.27).

11.2.2.9 Beobachtung 15: Suche nach Template V

Den Abschluss dieser empririschen Studie bildet ein Problem, das in gleicher Form bereits in einer anderen Sitzung beobachtet werden konnte: Eine Datei soll zeilenweise gelesen und dabei in ein Array (vom Basistyp `String`) übertragen werden (s. Kap.11.2.1.5). In der Sitzung S_{77} kann die Hinführung zu diesem zum Zeitpunkt t_{656} festgemacht werden: T_{288} reaktiviert die Phase der Arbeitsvorbereitung[15] (*„Zusammenfassend wer macht denn jetzt das einlesen der datei zeilenweise in ein string array?"*), woraufhin sich T_{316} bereit erklärt, sich der genannten Teilaufgabe anzunehmen (t_{662}: *„Wenn ich die information darüber gleich finde Dann kann auch ich das übernehmen."*).

Sechs Minuten später stösst T_{316} auf Schwierigkeiten bei der Umsetzung und wendet sich damit an die Gruppe (t_{865}: *„Ich bin nicht sicher wie kann man das ganze in string array packen."*), um bereits kurze Zeit später Unterstützung von Teammitglied T_{288} in Form eines Beispiels zu erhalten(t_{1042}: *„Zusammenfassend kann man in untiteld4 sehen wie zeilenweises einlesen funktionieren soll."*). Dieses Beispiel ist nachfolgend wiedergegeben.

Beispiel 11.10: Quellcode des Lösungsvorschlags (Problem 15)

```
//Zeilenweise lesen aus einer Datei test.dat
import java.io.*;
public class ZeileEinlesen
{
  public static void main (String args[])
  {
    BufferedReader in = new BufferedReader(
                            new InputStreamReader(
                                new FileInputStream("testdat.txt")));
    while (true)
    {
      // Einlesen einer Zeile in die Variable s
      String s=in.readLine();
```

[14] Benutzeraktivitäten, die nicht mittels der VitaminL-IDE durchgeführt werden, können derzeit nicht vom System erfasst werden. Dazu zählen unter anderem externe Aktivitäten wie die Literaturrecherche.

[15] Konkreter: der Aufgabenverteilung

```
      // wenn s keinen Wert hat ist das File zu Ende
      if (s==null) break;
      // ausschreiben von s
      System.out.println(s);
    }
    }
    // catch wird ausgeloest, wenn es einen Fehler unter try gibt
    catch (Exception e)
    {
      System.out.println(e); //Ausschreiben der Fehlermeldung
    }
  }
}
```

■

Dieser Ansatz wird von T_{290} noch um einen wichtigen Aspekt ergänzt (t_{1088}: *„Ich denke Du musst zuerst herausfinden wie viele zeilen es gibt, und erstellst dann ein Array entsprechend dieser Größe"*), so dass T_{316} im Prinzip alle für eine erfolgreiche Bearbeitung seiner Teilaufgabe notwendigen Informationen zur Verfügung stehen.

Im weiteren Verlauf der Sitzung kann über einen Zeitraum von zunächst fünf Minuten zunächst eine Vielzahl von Dokumentenwechseln seitens T_{316} festgestellt werden (s. Tab.E.29, $t_{1094} - t_{1342}$), bevor dieser Teilnehmer mit der Bearbeitung seiner Quelltexte fortfährt ($t_{1366} - t_{1957}$). Eine weitere Anhäufung kann im Anschluss daran beobachtet werden ($t_{1959} - t_{2175}$), unmittelbar bevor T_{316} das von T_{288} präsentierte Beispiel anfordert (t_{2177}: *„Zur Ausarbeitung ich brauche Untitled4."*). Unter Berücksichtigung eines weiteren Lösungsvorschlags von T_{288} (t_{2315}: *„Zusammenfassend LineNumberInputStream lsis = new LineNumberInputStream(new FileInputStream(file)); while ((c = lsis.read())!= -1) ; System.out.print(lsis.getLineNumber());"*) entwickelt T_{316} schließlich den in Beispiel 11.11 dargestellten Quellcode und stellt ihn seinen Teammitgliedern zur Diskussion.

Beispiel 11.11: Weiterentwickelter Quellcode (Problem 15)

```
import java.io.*;

public class ReadAndWrite
{
  //LineNumberReader f;
  //String line;
  String feld [] = new String [24];
  BufferedReader in=new BufferedReader(
                        new InputStreamReader(
                            new FileInputStream("namen.txt")));
  for ( int i =0;  i<24; i++)
  {
    String[i] = in.readLine();
    /* try {
       f = new LineNumberReader(
           new FileReader("namen.txt"));
       while ((line = f.readLine()) != null) {
         System.out.print(f.getLineNumber() + ": ");
         System.out.println(line);
       }
```

```
      f.close();
    } */
  }
  catch (IOException e)
  {
    System.out.println("Fehler beim Lesen der Datei");
  }
}
```

■

Auch hier kann sehr gut die bereits bei Problem 13 (s. Kap.11.2.2.7) festgestellte Vorgehensweise beobachtet werdfen, gemäß welcher Lösungen – auf der Basis vorhandener und als geeignet erscheinender Beispiele – schrittweise im Dialog mit den Teammitgliedern entwickelt werden. Ein Eingreifen eines Tutors würde diesen Vorgang sicher zeitlich beschleunigen, jedoch ist von solch einem Eingreifen im Hinblick auf den gewünschten Lerneffekt eher abzuraten.

11.3 Ergebnisse

11.3.1 Mehrdeutigkeit von Problemsituationen

Zunächst einmal ist festzustellen, dass die Abbildung der dargelegten Problemsituationen auf die in Kapitel 11.1 vereinbarte Klassifikation nicht immer eindeutig erfolgen kann: Am Beispiel von Problem 9 wird zwar ein Bedienproblem festgestellt, dessen Ursachen können jedoch auch in einem fehlerhaften Verständnis des Erstellungsprozesses von Java-Programmen gesucht werden (s. Kap.11.2.2.3). Dies deckt sich mit den Erkenntnissen aus Kapitel 11.1.3. Infolgedessen ist die Analyse von Problemsituationen auch stets mit einem gewissen intellektuellen Aufwand behaftet: Gleichen Problemsituationen liegen nicht notwendigerweise identische (oder zumindest ähnliche) Sitzungsverläufe zugrunde, die sich beispielsweise in Form von bestimmten, wiederkehrenden CLS-Code-Sequenzen oder -Mustern äußern. Dies kann man leicht anhand des mehrfach auftretenden Problems nach der Suche eines geeigneten Templates (s. Kap.11.2.1.4, 11.2.1.5, 11.2.2.6, 11.2.2.8 und 11.2.2.9) unter Zuhilfenahme der zugehörigen Sitzungsausschnitte nachvollziehen.

Für die Erkennung von Problemsituationen bedeutet dies, dass der im Rahmen der Rollenanalyse verfolgte Ansatz, anhand des Gebrauchs bestimmter CLS-Codes auf eine bestimmte Rollenausprägung zu schließen, sich nicht auf die Problemanalyse übertragen lässt: Die Entwicklung eines Verfahrens zur Problemanalyse, welches in der Lage ist, anhand des Auftretens eines (oder mehrerer) bestimmter CLS-Codes auf das Eintreten oder Vorhandensein einer Problemsituation einer spezifischen Kategorie zu schließen, kann aufgrund der in Kapitel 11.2 erhobenen Fälle weitestgehend ausgeschlossen werden. Ungeachtet dessen lassen sich aus den beobachteten Sitzungen potentielle Indikatoren ableiten, mit deren Hilfe allgemein auf die Existenz von Problemsituationen geschlossen werden kann.

11.3.2 Problemindikatoren

Diese Indikatoren verstehen sich nicht als ein sicheres Signal auf akute Probleme, sondern vielmehr als Hinweis auf mögliche Probleme: Das Auftreten eines solchen Indikators deutet nicht zwingend auf ein Problem hin, dennoch kann dieser in Problemsituationen oftmals beobachtet werden, so dass er als Ausgangspunkt für eine vertiefende Analyse angesehen werden sollte.

11.3.2.1 Ausbleibende Reaktionen

Wie bereits in GÖLDNER (2005) dargelegt wurde, ist speziell die Kommunikation während der gemeinsamen synchronen Programmierung einerseits unerlässlich, andererseits aber auch anfällig gegenüber einer Vielzahl von Störeinflüssen und weiteren Effekten, die sich nachteilig auf den Gruppenprozess auswirken können (vgl. GÖLDNER, 2005). Insbesondere das Ausbleiben von Reaktionen anderer Gruppenmitglieder (vgl. GÖLDNER, 2005, S.59f) konnte in den zur Problemanalyse herangezogenen Sitzungen im Kontext von Problemsituationen oftmals beobachtet werden. So wendet sich beispielsweise im Kontext von Problem 5 der Teilnehmer T_{304} mit einer Fragestellung an sein Team (s. Tab.E.9, t_{927}: *„Entschuldigt mich wie heißt denn z.B. das Array was ich sortieren soll? leseDatei[ ] oder wie?"*), erhält aber im weiteren Verlauf der Sitzung keine Antwort (s. Tab.E.9) und wendet sich darauf hin einem anderen Teilaspekt der Aufgabenstellung zu; sein ursprüngliches Problem bleibt unbearbeitet. Ein weiteres Beispiel kann dem Umfeld von Problemsituation 8 entnommen werden, in dessen Verlauf sich T_{293} zum Zeitpunkt t_{2009} ebenfalls mit einer Frage an ein Gruppenmitglied (T_{292}) wendet, welches sich jedoch seinerseits gerade mit einer anderen Fragestellung an Teilnehmer T_284 als drittem Teammitglied wendet (s. Tab.E.14). In der Folge bleibt die Frage von T_{293} unbeantwortet.

11.3.2.2 Wiederholung von Beiträgen

Ein weiteres bereits bei GÖLDNER (2005) diskutiertes Kommunikationsproblem stellt die Beitragswiederholung dar, die sich auf fehlende Rückmeldungen der übrigen Teammitglieder hinsichtlich des originären Beitrags zurückführen lassen vgl. GÖLDNER, 2005, S.61. Dies zeigt sich unter anderem in Problemsituation 15, die mit einer durch Teilnehmer T_{288} initiierten Aufgabenverteilung beginnt (t_{656}: *„Zusammenfassend wer macht denn jetzt das einlesen der datei zeilenweise in ein string array?"*). Nachdem unmittelbare Reaktionen der anderen Teammitglieder ausbleiben, wiederholt T_{288} nach 91 Sekunden seinen Beitrag und wählt dabei eine etwas andere Formulierung (t_{659}: *„Zusammenfassend hat also keiner lust das zu übernehmen?"*) und erhält daraufhin eine zufriedenstellende Reaktion eines Teammitglieds (t_{662}; s. Tab.E.28). Somit hat sich die Strategie der Beitragswiederholung in dieser Problemsituation als erfolgsbringend erwiesen.

Ein relativ ungewöhnliches Beispiel einer solchen Wiederholung findet sich zu Beginn von Problemsituation 5 (s. Tab.E.9): Teilnehmer T_{304} beginnt die Sitzung mit

dem Versuch einer Aufgabenverteilung (t_4: *„Zusammenfassend also wer will was machen?"*), die umittelbar darauf (t_5) im Abstand von nur 23 Sekunden wiederholt wird. In dieser sehr kurzen Zeitspanne kann jedoch nicht ernsthaft mit einer verwertbaren Rückmeldung seitens der anderen Teilnehmer gerechnet werden.

Eine weitere Beitragswiederholung ist im Kontext von Problemsituation 3 zu finden (s. Tab.E.5): Teilnehmer T_{292} wendet sich zum Zeitpunkt t_{42} mit einem Bedienproblem (*„Verzeihung aber wie füge ich jetzt das programm ein???"*;) an seine Gruppe, erhält aber nur eine unbefriedigende Antwort von T_{291} (t_{53}: *„Ich bin nicht sicher aber irgendwie kann man hier nichts einfügen."*) und wiederholt dann seine Frage, diesmal direkt an das dritte Teammitglied T_{293} gerichtet (t_{72}: *„Verzeihung wie füge ich das programm ein, es geht nicht."*). Dieser verweist auf den anwesenden (realen) Tutor (t_{94}: *„Ich bin ziemlich sicher das du da lieber T_{284} fragen solltest."*), was dazu führt, dass T_{292} seine Frage ein weiteres Mal wiederholt (t_{153}: *„Verzeihung wie kann ich hier programme aus dem internet einfügen?"*), was dann letztlich mit einer hilfreichen Antwort durch den Tutor belohnt wird.

Im Kontext von Problem 13 findet ebenfalls eine Wiederholung eines Beitrags statt, der jedoch nicht auf ein Problem hinweist, sondern auf eine mögliche Problemlösung: Zum Zeitpunkt t_{263} präsentiert Teilnehmer T_{288} seinen Teammitgliedern einen Lösungsvorschlag (*„Lasst mich Euch zeigen hier steht was für uns: Listing1809.java"*), den er in anderer Formulierung knapp zwei Minuten später wiederholt (t_{572}: *„Lasst mich Euch zeigen guckt euch mal meine klasse an..."*), da Reaktionen seiner Teammitglieder zunächst ausbleiben (s. Tabelle E.23).

Problematisch an diesem Indikator sind zwei Umstände, die seine Tauglichkeit für Zwecke der Rollenanalyse herabsenken:

1. Es muss durch das System erkannt werden, dass ein Beitrag eines Teilnehmers eine Wiederholung eines zuvor von ihm getätigten Beitrags darstellt. Dies wird umso schwerer, je mehr sich die Beiträge in ihren Formulierungen voneinander unterscheiden (siehe dazu die Ausführungen zu den Problemsituation 3 und 15).

2. Die Wiederholung von Beiträgen kann – wie am Beispiel der Problemsituation 13 dargestellt – auch eingesetzt werden, um Aufmerksamkeit von den übrigen Teammitgliedern zu erlangen, ohne, dass direkt ein Problem vorliegt.

11.3.2.3 Fragen

Naturgemäß lassen einige Kommunikationsbeiträge eher auf Problemsituationen schließen als andere: Insbesondere Fragen wie beispielsweise *„Weisst Du...?"* (CLS-Code 11), *„Kannst Du mir mehr sagen...?"* (CLS-Code 12) oder auch *„Kannst Du erläutern, warum/wie...?"* (CLS-Code 13) deuten darauf hin, dass beim Fragesteller als Auslöser des entsprechende Kommunikationsbeitrags ein Klärungsbedarf hinsichtlich eines gegebenen Sachverhalts besteht. Beispiele dafür finden sich in den diskutierten Problemsituationen zahlreich (s. t_{802} in Tab.E.2, t_{2789} in Tab.E.19 oder auch t_{2360} in Tab.E.28).

Ergänzend gibt es auch Kommunikationsbeiträge, die keine Frage im eigentlichen Sinne darstellen, aber dennoch auf ein Informationsdefizit hindeuten. Beiträge wie *„Bitte zeige mir..."* (CLS-Code 16) oder auch *„Ich bin nicht sicher..."* (CLS-Code 24) sind typische Vertreter dieser Kategorie (s. t_{3107} in Tab.E.9 bzw. t_{865} in Tab.E.28).

Prinzipiell zeigt sich eine Schwierigkeit bei der Bewertung, ob es sich nun tatsächlich um eine Problemsituation handelt, in welcher die Gruppe oder der einzelne Teilnehmer Unterstützung benötigt, oder ob eine Frage innerhalb einer Gruppendiskussion zur Problemlösung ohne den unterstützenden Eingriff eines Tutors führt. Erschwerend kommt der zum Teil sinnentfremdende Gebrauch der diversen CLS-Elemente hinzu, der einer Problemanalyse allein auf Basis der verwendeten CLS-Code entgegensteht. So verwendet beispielsweise Teilnehmer T_{286} in der Problemsituation 4 (s. Tab.E.7) bei insgesamt 36 Kommunikationsbeiträgen 29 mal den CLS-Code 27 (*„Ich denke..."*), darunter finden sich auch solche Beiträge wie *„Ich denke danke."* (t_{1005}) oder auch *„Ich denke ich bin dumm, kann mir mal bitte wer erklären wieso ich das erste mal auf datei zugreifen kann und er beim zweiten mal meint, as Datei evtl. nicht initialisiert wurde?"* (t_{4460}). Insbesondere bei diesem zweiten Beispiel wird eine als Vorschlag verstandene Äußerung in eine Fragestellung überführt und damit die mit dem CLS-Code 27 assoziierte Intention eines Vorschlags grundsätzlich verfälscht. Ein ähnliches Verhalten findet sich auch in Problemsituation 5 (s. Tab.E.28): Dort ist zu beobachten, dass Teilnehmer T_{288} offenbar eine Vorliebe für CLS-Code 2 entwickelt hat, denn eine Vielzahl seiner Beiträge beginnen mit dem zum Code 2 zugehörigen Satzanfang *„Zusammenfassend..."*, wobei das als Zusammenfassung vereinbarte Kommunikationselement auch für Fragestellungen (t_{656}: *„Zusammenfassend wer macht denn jetzt das einlesen der datei zeilenweise in ein string array?"* oder auch t_{659}: *„Zusammenfassend hat also keiner lust das zu übernehmen?"*) zum Einsatz kommt. Dieser bevorzugte Einsatz bestimmter Kommunikationselemente des CLS-Ansatzes können auch in vielen anderen der eingangs diskutierten Problemsituationen festgestellt werden. In der Folge treten Beiträge mit zum Teil sehr verwirrenden Satzkonstruktionen zutage (s. Tab.E.11, t_{3691}: *„Deshalb , ich dahcte das letzte mal auch, aber musste ich noch angeben."* oder Tab.E.19, t_{5565}: *„Ich denke, wir sollten du kannst n char in nen int umwandeln."*). Diese werden zwar in der Regel von den (menschlichen) Teammitgliedern verstanden, ein (virtueller) Tutor wird in so einem Fall jedoch vor eine recht komplexe Aufgabe gestellt.

11.3.2.4 Navigationsoperationen

Wie bereits bei den jeweiligen Problemsituationen ausgeführt wurde, konnten bei einigen Problemen Anhäufungen von Navigationsoperationen identifiziert werden. Es handelt sich dabei um die CLS-Codes 35 (Dokumentenwechsel) und 36 (Zeilenwechsel innerhalb eines Dokuments). Es ergibt sich daraus ein interessanter Ansatzpunkt für die weitere Analyse zur Erkennung von Problemsituationen, bei dem jedoch weitere Aspekte zu berücksichtigen sind. Zum einen muss beim Navigieren zwischen zielgerichteter und zielloser Navigation differenziert werden: Während

erstere in einem Szenario mit mehreren offenen Quelltextdokumenten, wie es in dem durch VitaminL unterstützten Lernprozess üblich ist, zur Informationsgewinnung für eine erfolgreiche Problemlösung unerlässlich ist, tritt die zweite Variante in Problemsituationen auf. Mit dem ziellosen Navigieren wird kein Plan im Sinne einer erfolgreichen Problemlösung verfolgt, es erscheint vielmehr wie eine Art Verlegenheitsgeste in Ermangelung einer sinnvollen Strategie für das weitere Vorgehen. Darüber hinaus werden Navigationsoperationen mit CLS-Code 36 auch beim Bearbeiten eines Quelltextes durch Zeilenwechsel generiert. Hier muss noch eine Verbesserung der VitaminL-IDE erarbeitet werden, die diese speziellen Operationen identifiziert und herausfiltert, so dass innerhalb des Verfahrens zur Problemanalyse differenziert werden kann zwischen solchen Zeilenwechseln, die als Folge von Editoroperationen (wie Einfügen und Löschen von Quelltext) entstehen, und solchen, die vom Benutzer tatsächlich durch Ansteuern einer neuen Zeile in einem Quelltextdokument generiert werden – und zwar muss dieses geschehen, ohne dass die Funktionsweise des Quelltexteditors der VitaminL zu beeinträchtigt wird.

11.3.2.5 Inaktivitäten

Das Problem des inaktiven Teilnehmers wurde bereits im Rahmen des sozialen Prozesses als Teil des Vorgehensmodells diskutiert (s. Kap.9.1.2). Auch in den hier betrachteten Problemsituationen konnten mehrfach Zeiträume identifiziert werden, in denen das System keine Aktionen einzelner Teilnehmer verzeichnen konnte: Aus der Sicht des Systems – und damit unter Umständen auch aus Sicht des virtuellen Teams und seiner Mitglieder – muss solch ein Teilnehmer dann als inaktives Teammitglied angesehen werden. Ein Beleg dafür findet sich beispielsweise im Kontext von Problemsituation 1: Zwischen den Zeitpunkten t_{1753} (0:47:47) und t_{1857} (0:49:28) werden von Teilnehmer T_{305} zunächst mehrere Navigationsoperationen registriert, an die sich eine mehr als drei Minuten andauernde Phase der Inaktivität anschließt, bis sich T_{305} schließlich von seinem aktuellen Problem abwendet und dies somit offen und unerledigt bleibt (t_{2039}: *„Deshalb höre ich jetzt auch auf zu suchen und werde konstruktiv. habt ihr bestimmte sachen zu deligieren vielleicht? im Übrigen: T_{303}, in setTitle muss ne variable, weil da das gewählte getränk hin soll..."*)[16].

Bei einem Analyseansatz, der auf der Protokollierung von Zeiträumen beruht, in welchen einem Teilnehmer keine Benutzeraktionen zugeordnet werden können, muss berücksichtigt werden, dass Phasen der Inaktivität aus Systemsicht nicht generell mit Phasen der Inaktivität gleichzusetzen sind. So gibt es durchaus Aktivitäten, die einen wichtigen Beitrag zur Aufgabenbearbeitung oder Problemlösung beitragen, aber außerhalb der VitaminL-IDE durchgeführt werden und von dieser daher nicht erfasst werden können. Die Recherche in Vorlesungsskripten oder im Internet zum Zwecke der Informationsbeschaffung stellt solch eine Aktivität dar. So werden beispielsweise in Problemsituation 11 vom Teilnehmer T_{294} zwischen

[16] Das Verfassen solch eines Beitrags wird durch das System nicht erfasst, erfordert aber selbstverständlich auch etwas Zeit, die in diesem Fall von den erwähnten drei Minuten der Inaktivität abzuziehen sind.

den Zeitpunkten t_{1676} (0:29:35) und t_{2044} (0:33:54) keinerlei Aktionen registriert (s. Tab.E.20). Aus Systemsicht stellt dies eine Phase der Inaktivität von fast viereinhalb Minuten dar. Aus dem nachfolgenden Kommunikationsbeitrag (*„Zusammenfassend ich auch nicht, in er api steht leider auch nix, zumindest hab ich noch nix gefunden."*) kann geschlossen werden, dass T_{294} diese Zeit zur Informationsbeschaffung genutzt hat. Im Anschluss an diesen Beitrag folgt eine weitere Phase von fast sechs Minuten (bis t_{2044} = 0:39:49), in welcher das System keine Benutzeraktionen von T_{294} verzeichnen kann. Es kann nur gemutmaßt werden, dass auch dieser Zeitraum für Recherchearbeiten genutzt wird. Erschwerend für die Nutzung des Indikators *Inaktivität* für die Problemanalyse kommt hinzu, dass es auch Phasen der Inaktivitäten gibt, die sich aus diversen Gründen als erforderlich erweisen, aber – im Gegensatz zur Informationsbeschaffung – keinen direkten Beitrag zur Aufgabenbearbeitung leisten (s. den Beitrag von T_{305} zum Zeitpunkt t_{1032} in Tab.E.9).

11.3.3 Folgerungen für die Problemidentifikationskomponente

Die bis dato gewonnenen Ergebnisse sind nicht ohne Auswirkung auf die geplante Komponente zur Problemidentifikation, die als eines der Kernelemente innerhalb des virtuellen Tutors maßgeblich verantwortlich ist für das Auslösen von Supportanforderungen in Problemsituationen (s. Kap.8.5, Abb.8.29 und 8.30).

11.3.3.1 Automatische Detektion von Problemsituationen

Die Problemidentifikationskomponente sieht sich demnach bei der automatischen Erkennung von Problemen während einer Programmiersitzung mit einer Vielzahl von Schwierigkeiten und Hindernissen konfrontiert, die bei der Konzeptionierung und Umsetzung dieser Komponente entsprechende Berücksichtigung finden müssen:

1. Aufwand:
   Die Kategorisierung der untersuchten Probleme unter Zuhilfenahme der in Kapitel 11.1 vereinbarten Taxonomie erfolgt zum Teil mit nicht unerheblichem intellektuellen Aufwand.

2. Mehrdeutigkeit:
   Erschwerend kommt hinzu, dass bei der Klassifizierung ein gewisser Handlungsspielraum besteht, in Folge dessen die Probleme nicht immer eindeutig klassifiziert werden können.

3. Unvollständigkeit:
   Die im Rahmen der vorliegenden Studie erfassten Probleme decken die Problemkategoien nicht vollständig ab: Zwar konnten einige Probleme mehrfach beobachtet werden, dennoch traten in den der Studie zugrundeliegenden Sitzungen lediglich elf unterschiedliche Probleme auf. Diesen stehen 59 mögliche

Probleme des fachlichen Prozesses[17] sowie fünf Probleme des sozialen Prozesses gegenüber.

4. Unvergleichbarkeit:
   Probleme derselben Kategorie konnten nicht anhand vergleichbarer Sequenzen oder Muster von CLS-Codes identifiziert werden. Obwohl fünf Problemsituationen sich mit der Suche nach einem geeigneten Template befassten, weisen die entsprechenden Sitzungen keine Gemeinsamkeiten hinsichtlich der Verwendung bestimmter CLS-Codes auf, die diese eindeutig von anderen Problemsituationen unterscheiden würden. Dies trifft sowohl auf einzelne Codes als auch auf Sequenzen oder Muster mehrere CLS-Codes zu.

Dem gegenüber stehen die genannten Problemindikatoren, die einige interessante Ansätze bieten, mit denen zumindest eine Analyse hinsichtlich des prinzipiellen Auftretens von Problemen realisiert werden kann. Eine genauere Klassifikation gemäß der vereinbarten Problemkategorien wird dabei jedoch ausgeschlossen. Hier bieten sich ergänzende Strategien an, die die automatische Problemerkennung innerhalb einer Tutor-Komponente sinnvoll erweitern.

11.3.3.2 Unterstützungsstrategien

11.3.3.2.1 Analyse von Nachrichteninhalten Die bisherigen Betrachtungen stützen sich – analog zur Rollenanalyse – vornehmlich auf die CLS-Codes der auf dem Server – und damit auch in der Tutorkomponente – eingehenden Nachrichten. In Anbetracht der bisher erzielten Ergebnisse empfiehlt es sich, das Verbesserungspotential zu untersuchen, das sich bei Ausdehnung des code-basierten Analyseansatzes unter Berücksichtigung weiterer Informationen ergibt.

Bereits jetzt sind mit jeder eingehenden Nachricht – neben der Angabe des nachrichtenauslösenden Benutzers sowie deren CLS-Code – der Zeitpunkt des Auslösunes und Versendens einer Nachricht[18] sowie – je nach Nachrichtentyp beziehungseise zugehöriger Operation – unter anderem Angaben zum betroffenen Dokument (Dokumentenoperationen) oder Kommunikationsbeiträge (Kommunikation)[19].

Wie der vorliegenden Studie entnommen werden kann, liefern neben den Botschaften mit CLS-Code 47[20] insbesondere die Inhalte der Kommunikationsbeiträge

[17] 33 Probleme ergeben sich bei rein prozessorientierter Sichtweise, weitere 26 Probleme kommen bei fehlerorientierter Sichtweise hinzu (s. Kap.11.1.1

[18] Da alle Sitzungen dieser Studie in einem privaten LAN stattfanden, sind die Transportzeiten zwischen Client und Server vernachlässigbar gering, so dass die angegebenen Zeiten auch als Zeitpunkte für den Nachrichtenempfang auf dem Server angesehen werden dürfen.

[19] Detaillierte Angaben zu Aufbau und Inhalt der diversen Nachrichtenklassen sind in Kapitel 8.4.2 zu finden.

[20] Hinter CLS-Code 47 verbirgt sich eine der beiden derzeit verfügbaren Java-Operation, nämlich der Aufruf des Compilers zur Übersetzung eines Quelltext-Dokuments. Mit dieser Botschaft wird das Ergebnis des Übersetzungsvorgang (insbesondere resultierende Fehlermeldungen des Compilers) vom Client zum Server transportiert.

oftmals wertvolle Hinweise auf Probleme. Die Auswertung dieser und weiterer verfügbarer Informationen bietet sich folglich zur Ergänzung einer rein codebasierten Problemanalyse an.

11.3.3.2.2 Benutzerinitiierte Hilfeanforderung Wie aus der vorliegenden Studie ersichtlich ist, sind die Teilnehmer durchaus bereit und in der Lage, sich bei bestehenden Problemen nicht nur an ihre Teammitglieder zu wenden, sondern auch (scheinbar virtuelle) Tutoren zu konsultieren. Hierbei hat es sich gezeigt, dass dem Tutor auch ergänzende Informationen beziehungsweise konkrete Fragestellungen übermittelt werden (s. z.B. Tab.E.5, t_{947}, Tab.E.15, t_{3992} oder Tab.E.17 , t_{5674}).

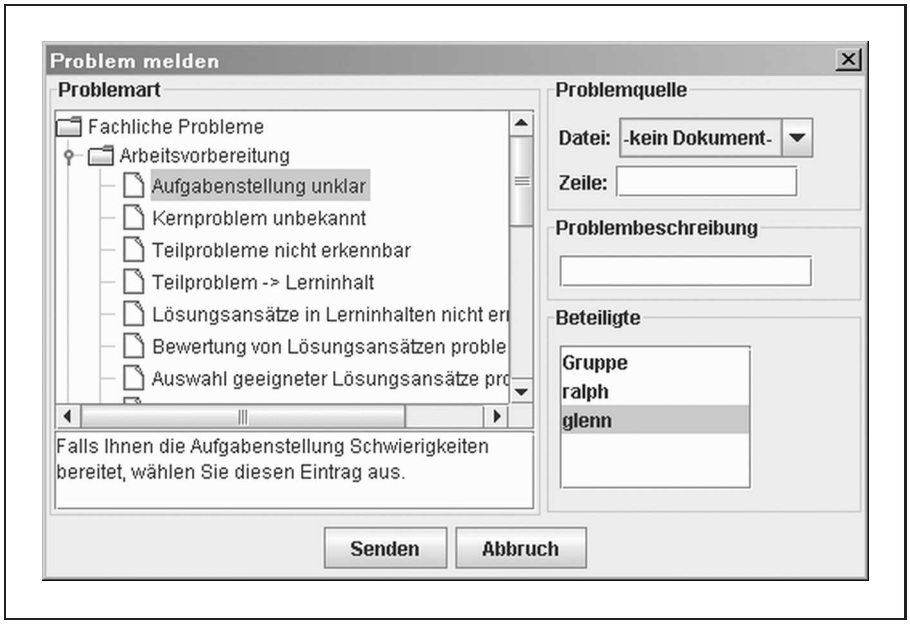

Abb. 11.1: Dialog für benutzerinitiierte Problemmeldungen
(Screenshot)

Dieser Ansatz führt in Konsequenz zu einer Ergänzungsstrategie innerhalb der Tutorkomponente, bei welcher der Benutzer im Bedarfsfall den (virtuellen) Tutor konsultiert und diesen im Rahmen solch einer Anfrage neben einer möglichst genauen Einschätzung hinsichtlich der Problemkategorie mit geeigneten Zusatzinformationen versorgt. Somit geht mit diesem Ansatz auch eine inhaltsbasierte Problemanalyse einher, wie sie bereits im vorigen Kapitel diskutiert wurde. Die Abbildung 11.1 zeigt an einem konkreten Prototypen, wie die Implementierung eines entsprechenden Dialogfensters innerhalb des VitaminL-CLient zur benutzeriniitierten Problemmeldung respektive Unterstützungsanforderung gestaltet werden kann.

Neben dem Auslöser einer Problemmeldung sowie deren Zeitpunkt werden über den Dialog weitere Informationen erfasst und der Tutorkomponente übermittelt:

1. Problemart:
   In einem Auswahlbaum werden die in Kapitel 11.1 vereinbarten möglichen

Probleme gemäß der dort getroffenen Kategorisierung hierarchisch dargestellt. Somit kann der Benutzer je nach Situation zwischen einer exakten Einschätzung der jeweiligen Problemkategorie (zum Beispiel Kategorie A.1: *„Aufgabenstellung unklar"*) und einer nur sehr groben Auswahl (beispielsweise *„Fachliche Probleme"*) auswählen. Auf diese Weise wird dieses Verfahren auch unterschiedlichen Kenntnisständen beziehungsweise Erfahrungshorizonten der Teilnehmer gerecht. Unterstützt wird die Auswahl durch eine textuelle Erläuterung unterhalb des Auswahlbaums, die weitere Informationen über die derzeit ausgewählte Problemart anbietet. Darüber hinaus gibt es auf jeder Hierarchiebene des Auswahlbaums einen Eintrag *„Sonstiges"*, um Benutzern auch dann die Möglichkeit einer Problemmeldung zu geben, wenn es ihnen nicht möglich ist, das aktuelle Problem zu klassifizieren.

2. Problemquelle:
   Hier hat der Problemauslöser die Möglichkeit, Angaben zur Lokalität des aufgetretenen Problems zu tätigen, indem er eines der geöffneten Dokumente auswählt und bei Bedarf noch weitere Angaben zur Zeilennummer macht. Weitere Auswahlmöglichkeiten sind *„kein Dokument"*, falls keines der Dokumente für das aktuelle Problem relevant sind, oder auch *„keine Angabe"*, falls die Benennung eines betroffenen Dokuments nicht möglich oder nicht sinnvoll ist.

3. Problembeschreibung:
   In einem kleinen Eingabefeld kann der Benutzer bei Bedarf weitere Informationen zum aktuellen Problem hinterlassen. Diese Eingabe ist derzeit keinerlei Einschränkungen unterlegen.

4. Beteiligte:
   Mittels einer Auswahlliste kann der problemauslösende Benutzer die am aktuellen Problem beteiligten Mitglieder auswählen und der Tutorkomponente als zusätzliche Information übermitteln. Dabei besteht die Möglichkeit, neben einem weiteren Mitglied auch mehrere Sitzungsteilnehmer zu selektieren. Ferner steht der Eintrag *„Gruppe"* als Auswahlmöglichkeit zur Verfügung, der sich dann anbietet, wenn die gesamte Gruppe von einem Problem betroffen ist.

Dieses derzeit – als Ergänzung des bestehenden VitaminL-Systems – nur prototypisch umgesetzte Konzept (s. Abb.11.1) ist in weiteren Varianten denkbar, die sich hinsichtlich der Art der Anforderung tutorieller Unterstützung sowie der an die Tutorkomponente übermittelten Informationen vod diesem Prototypen unterscheiden:

1. Fragestellungen:
   Ähnlich wie die bereits beobachtete Vorgehensweise einiger Probanden ist ein Ansatz denkbar, welcher einen Kommunikationsbeitrag mit CLS-Code 34 (*Hilfe durch Tutor*) um eine konkrete Fragestellung (z.B. *„Kannst Du mir*

mehr sagen bzw. mal ein beispiel geben zur verwendung von „instance of"?"; s. Tab.E.2, t_{802}) ergänzt.

2. Stichwörter:
   Anstelle kompletter Fragestellungen ist auch ein Konzept vorstellbar, bei welchem Kommunikationsbeiträge mit CLS-Code 34 um ein oder mehrere Stichwörter ergänzt und als eine Art Suchanfrage an die Tutorkomponente übermittelt werden. So könnte beispielsweise eine Anfrage wie *„String Sortieren"* ausgelöst werden, um Hilfe bei der Sortierung von String-Objekten zu erhalten, während *„Instanz Skript"* im Idealfall diejenigen Stellen im Vorlesungsskript benennt, die sich mit Instanzen (von Klassen) befassen.

Ferner ist ein Verzicht auf die Problemklassifikation durch den Benutzer diskutierbar: Speziell Anfänger besitzen in der Regel nur geringe Vorkenntnisse, so dass Klassifikationsversuche konkreter Probleme unter Umständen gänzlich misslingen. In der Folge wird die Tutorkomponente mit fehlerhaften Informationen versorgt. Solche Fehleinschätzungen oder Fehlklassifikationen sind daher mit Vorbehalt zu betrachten. Ein Verzicht auf diese würde in so einem Fall die bessere Wahl darstellen: Die Arbeitsweise der Tutorkomponente würde statt auf fehlerhaften *nur* auf fehlenden Informationen basieren.

11.3.3.2.3 Berücksichtigung des Vorgehensmodells Bereits im Rahmen der Rollenanalyse wurden Möglichkeiten sowie Vor- und Nachteile diskutiert, die aus der Anwendung des in Kapitel 9.1 erarbeiteten Vorgehensmodells auf VitaminL-Sitzungen resultieren. Dieser Ansatz bedeutet – übertragen auf die Problemanalyse – eine Reduzierung möglicher Problemkategorien in konkreten Problemsituationen: Eine Strukturierung von VitaminL-Sitzungen auf der Basis des VitaminL-Vorgehensmodells stellt nicht nur eine Art Rahmen dar, in welchem sich die Teilnehmer während der Aufgabebearbeitung bewegen, sondern begrenzt auch – je nach aktuellem Fortschritt innerhalb einer Sitzung – die relevanten Problemsituationen, die innerhalb einer Phase auftreten können.

Beispielsweise lassen sich in der Phase *Problemerkennung* (als Bestandteil des Teilprozesses *Arbeitsvorbereitung*) Probleme nachfolgender Phasen und Teilprozesse des fachlichen Prozesses ausschließen, so dass insgesamt eine Fokussierung auf diejenigen Problemkategorien stattfinden kann, die der jeweils aktuellen Einheit des Vorgehensmodells (hier: Problemerkennungsphase) oder vorgelagerten Einheiten zuzuordnen sind. In letztgenannten Fall ist zu diskutieren, ob ein Rücksprung in die entsprechende Einheit des Vorgehensmodells anzuraten ist.

11.3.3.3 Konzept zur technischen Umsetzung

Die Problemanalyse basiert – ähnlich der Rollenanalyse – auf der Verarbeitung eingehender Nachrichtenobjekte als Repräsentanten durchgeführter Benutzeraktionen. Dementsprechend knüpft eine technische Umsetzung der Problemanalyse

nahtlos an die in Kapitel 10.3.2 skizzierte Umsetzung der Rollenanalyse an: Sämtliche auf dem Server eintreffende Nachrichten eines virtuellen Teams werden – wie in Beispiel 8.8 skizziert (s. Kap.8.5.2) – an die Tutorkomponente zur genaueren Analyse weitergereicht und dort sowohl zur Bestimmung von Rollenausprägungen der Teilnehmer (s. Kap.10.3.2) als auch zur Identifikation von Problemsituationen verwendet.

Beispiel 11.12: Verarbeitung von Nachrichtenobjekten in der Tutorkomponente

```
void VTutor::analyzeMessage( VMessage msg )
{
  // Uebergib das Nachrichtenobjekt msg an die im Lernermodell
  // umgesetzte Komponente zur Rollenanalyse
  lernermodell.analyzeMessage( msg );

  // Uebergib das Nachrichtenobjekt msg ferner an die im Tutormodell
  // umgesetzte Komponente zur Identifikation von Problemen
  tutormodell.analyzeMessage( msg );

}
```

■

Die Problemerkennung gestaltet sich innerhalb des Tutormodells vielschichtig und besitzt neben einer Problemidentifikationskomponente pro Teammitglied eine weitere Analysekomponente, die auf die Erkennung von Problemen spezialisiert ist, welche das virtuelle Team als Ganzes (oder in Teilen) betreffen. Die Realisierung der einzelnen Teilkomponenten innerhalb des Tutormodells hängt letztlich stark von den Strategien und Konzepten ab, die für die Umsetzung ausgewählt werden. Hierbei ist die benutzeriniitierte Meldung von Problemsituationen besonders zu berücksichtigen, da diese einerseits konkrete Hinweise auf ein mögliches Problem beinhalten kann, die andererseits aber konträr zu den Ergebnissen der Tutorkomponente sein können, unter anderem bedingt durch mögliche Fehleinschätzungen des Benutzers.

Die Weiterreichung erkannter Probleme erfolgt innerhalb der Tutorkomponente an das Wissensmodell als nachgelagerte Subkomponenten zur Generierung eines passenden Unterstützungsangebots (s. Abb.8.29 und 8.30) in Form einer Kennzahl: Die in Kapitel 11.1 festgelegte Problemklassifikation wird mit einem Index versehen, so dass nicht nur jedes einzelne Problem, sondern auch alle übergeordneten Problemkategorien über einen eindeutigen Index identifizierbar sind. Dazu gehören letztlich auch zusätzliche Einträge wie *„Sonstiges"*, um dem Konzept benutzerinitierter Problemmeldungen Rechnung zu tragen (s. Kap.11.3.3.2.2). Zusammen mit dem Index werden weitere Informationen, die das aktuelle Problem betreffen, an das Wissensmodell als Unterstützungsgenerierungskomponente übergeben, sofern Zusatzinformationen verfügbar und notwendig sind. Insbesondere bedeutet dies auch eine Weitergabe der im Lernermodell enthaltenen Informationen in Form von aktuellen Rollenausprägungen der Teammitglieder, um ein Unterstützungsangebot zu generieren, das nicht nur an das jeweilige Problem angepasst ist, sondern auch die Zusammensetzung der vom Problem betroffenen Teilnehmer hinsichtlich des in Kapitel 9.2 festgelegten Rollenmodells berücksichtigt.

11.4 Zusammenfassung der Problemanalyse

Die Grundlage der Problemanalyse bildet die in Kapitel 11.1 vereinbarte Klassifikation, welche – in Anlehnung an das VitaminL-Vorgehensmodell (s. Kap.9.1) – potentielle Problemsituationen festlegt, die typisch für die synchrone Java-Programmierung in verteilten Teams sind. Im Rahmen einer Studie zur Vorbereitung der Konzeptionierung und Entwicklung einer Problemanalysekomponente wurden acht Sitzungen erfasst, an denen insgesamt 14 Studierende teilnahmen. In diesen Sitzungen konnten 15 Problemsituationen ausgemacht werden, die entweder durch die Teilnehmer selbst an das System gemeldet wurden oder durch einen bei allen Sitzungen anwesenden Tutor beobachtet wurden. Diese wurden im Hinblich auf ihre Ursachen und deren Identifizierbarkeit anhand der erfassten Benutzeraktionen ausführlich diskutiert.

Aufgrund der genannten Schwierigkeiten konnte jedoch kein allgemeingültiges Verfahren gefunden werden, das es erlauben würde, anhand der Benuzteraktionen, beziehungsweise der zugehörigen CLS-Codes, auf Probleme zu schließen und diese eindeutig anhand der gegebenen Problemtaxonomie zu klassifizieren. Stattdessen konnten einige Indikatoren identifiziert werden, bei deren Eintreten mit einer gewissen Wahrscheinlichkeit allgemein auf Probleme während der Aufgabenbearbeitung geschlossen werden kann. Diese Problemindikatoren bilden die Ausgangsbasis für eine Problemerkennungskomponente, zu deren Unterstützung eine Vielzahl weiterer Ansätze in die Konzeptionierung und Umsetzung einfließen kann. Insbesondere sind mit der Berücksichtigung des Vorgehensmodells, die auch schon im Rahmen der Rollenanalyse diskutiert wurde (s. Kap.10.5.1.3), und mit der Meldung von Problemsituationen durch die Benutzer zwei Strategien herausgearbeitet worden, deren Potential eine signifikante Verbesserung hinsichtlich der Erkennungsqualität der Problemanalysekomponente verspricht. Dieses zu belegen ist das Ziel von weiteren Untersuchungen, die sich an die vorliegende Arbeit anschließen.

Kapitel 12

Zusammenfassung

Lehrveranstaltungen zu den Themen *Programmierung* und *Software-Entwicklung* bestehen üblicherweise aus einer Vorlesung, in der per Frontalunterricht die Theorievermittlung stattfindet, und Praxiseinheiten, in denen die zuvor vermittelten Inhalte praktisch umgesetzt werden. Hierbei hat sich ein mehrstufiges Konzept aus aufeinander aufbauenden Einheiten bewährt. Diese Einheiten reichen von einfachen Beispielen und kleineren Übungsaufgaben, die in die Vorlesung eingebettet sind, über Tutorien und Hausaufgaben, in denen die Theorie anhand komplexerer, mehrere der zuvor behandelten Themen umfassenden Aufgaben vertieft wird, bis hin zum Abschlussprojekt, dessen Bearbeitung sich in der Regel über mehrere Wochen erstreckt und das eine Vielzahl der in der Lehrveranstaltung behandelten Sprachkonzepte abdeckt. Um die Vorteile kollaborativen Lernens nutzen zu können, erfolgt die Durchführung der Praxiseinheiten in Kleingruppen von üblicherweise drei Studierenden. Bei auftretenden Problemen können die Teilnehmer zu vereinbarten Zeiten die Unterstützung von Tutoren in Anspruch nehmen. Aufgrund der aufgezeigten Randbedingungen wie der begrenzten Verfügbarkeit der Tutoren, der räumlichen Verteilung vieler Studierender während der Abschlussarbeit und den auftretenden Problem empfiehlt sich der Einsatz eines ICLS zur Unterstützung virtueller Teams bei der Java-Programmierung.

Die Grundlage dieses ICLS bildet die VitaminL-IDE, die als CSCL-System mit einer Client-Server-Architektur eine Umgebung darstellt, in welcher sich virtuelle Teams zur gemeinsamen, zeitgleichen Bearbeitung von Programmieraufgaben mittels der objektorientierten Sprache Java treffen können. Die Kommunikation zwischen den Mitgliedern eines virtuellen Teams erfolgt mittels eines speziellen Chat in Form einer strukturierten Dialogschnittstelle, die ihrerseits auf den *Collaborative Learning Skills* von McManus & Aiken (1995) basiert. Die Kooperation, das heißt die gemeinsame, synchrone Erstellung und Bearbeitung von Java-Quelltexten, wird durch einen Gruppeneditor realisiert, welcher den Mitgliedern eines virtuellen Teams gemeinsame, verteilte Dokumente zur Verfügung stellt. Weitere Werkzeuge und Informationsdienste komplettieren den für die Programmierung erforderlichen Funktionsumfang der VitaminL-IDE. Sämtliche Aktionen (inklusive Kommunikationsbeiträge), die von einem Benutzer ausgelöst werden

können, werden durch Nachrichtenobjekte modelliert und sind zudem mit einer Codierung, den CLS++-Codes, versehen, die eine Erweiterung der CLS darstellen. Die technische Kommunikation zwischen Client und Server ist mit diesen Nachrichtenobjekten realisiert: Jede Benutzeraktion auf einem Client löst ein entsprechendes Nachrichtenobjekt aus, das über eine Netzwerkverbindung an den Server geschickt wird. Dort erfolgt die weitere Verarbeitung. Ergebnisse werden vom Server - ebenfalls in Form von Nachrichtenobjekten - an alle beteiligten Clients verteilt. Gleichzeitig werden alle auf dem Server eintreffenden Nachrichtenobjekte einer VitaminL-Sitzung ebendort vor ihrer eigentlichen Verarbeitung erfasst, in ein XML-Format überführt und in einer Protokolldatei gespeichert. Die Integration der geplanten Tutorkomponente – und damit verbunden der Ausbau des VitaminL-Systems von einer CSCL-Applikation zu einem ICLS – erfolgt ebenfalls server-seitig: Analog zur Nachrichtenprotokollierung werden sämtliche eingehenden Nachrichten an die Tutorkomponente weitergeleitet und dort weiterverarbeitet, um einerseits mit einer Problemanalysekomponente mögliche Problemsituationen zu erkennen und andererseits in einer Rollenanalysekomponente die Rollenausprägungen der einzelnen Teammitglieder zu bestimmen.

Das VitaminL-Rollenmodell – als ein Bestandteil des VitaminL-Arbeitsmodells – basiert auf dem Rollenmodell von Spencer & Pruss (1995) und resultiert aus dessen Adaption an das VitaminL-Vorgehensmodell. Dieses Vorgehensmodell stellt ein weiteres Element des VitaminL-Arbeitsmodells dar und bildet den organisatorischen Rahmen der Java-Programmierung: Strukturiert in Prozesse, Teilprozesse, Phasen und Schritte beinhalten die einzelnen Elemente des Vorgehensmodells sämtliche Tätigkeiten, die üblicherweise während der Aufgabenbearbeitung in der VitaminL-IDE durchlaufen werden. Aus der Zuordnung der Rollen nach Spencer & Pruss zu den Elementen des Vorgehensmodells resultieren die VitaminL-Rollen, die abschließend mit einer Gewichtung gemäß ihrer Relevanz für die Aufgabenbearbeitung und für den damit einhergehenden Lernprozess versehen werden.

12.1 Die Rollenanalyse und ihre Ergebnisse

Das Ziel einer an die jeweilige Teamzusammensetzung angepassten tutoriellen Unterstützung erfordert die Verfügbarkeit der Rollenprofile der einzelnen Teilnehmer innerhalb der Tutorkomponente bei Auftreten einer Problemsituation. Bislang müssen die Teilnehmer dazu vor ihrer ersten VitaminL-Sitzung einen aus 150 Elementen bestehenden, im Internet verfügbaren online-Fragebogen nach Spencer & Pruss ausfüllen, der anschließend in die Datenbank des VitaminL-Systems eingespeist wird. Die Zielsetzung der Rollenanalyse liegt in der Bereitstellung einer Software-Komponente, die – integriert in die Tutorkomponente – während einer laufenden Sitzung aktuelle Rolleninformationen der Teilnehmer generiert und damit den Fragebogen entbehrlich macht. Als Berechnungsgrundlage dienen die von den Teilnehmern ausgelösten Aktionen beziehungsweise deren CLS++-Codes. Die Rollenanalyse basiert auf der Annahme, dass eine linearer Zusammenhang zwi-

schen den Rollenausprägungen eines Teilnehmers und den von ihm verwendeten Aktionen existiert. In einer ersten Untersuchungsphase wurden daher protokollierte VitaminL-Sitzungen zusammen mit den Rollenprofilen der entsprechenden Teilnehmer mittels einer Regressionsanalyse auf die vermuteten Zusammenhänge hin untersucht. Die signifikanten Resultate dieser Phase wurden anschließend in einer prototypischen Analysekomponente umgesetzt und in die Tutorkomponente des VitaminL-Systems integriert. Anhand der Benutzertests einer zweiten Untersuchungsphase wurde die Rollenanalysekomponente evauliert: Aus den Benutzeraktionen wurden Rollenausprägungen berechnet und den korrespondierenden Fragebogenergebnissen gegenübergestellt. Die Resultate dieser Evaluierung belegen zunächst nicht die These, nach welcher sich aus den Benutzeraktionen eines Teilnehmers auf dessen Rollenausprägungen schließen lässt, da die Rollenausprägungen der Analysekomponente zum Teil erheblich von den zugehörigen Fragebogenwerten abweichen. Auch eine Regressionsanalyse, die auf die Sitzungsdaten und Rollenprofile der zweiten Phase angewendet wurde, liefert andere signifikante Ergebnisse als die Regressionsanalyse der ersten Phase. Mögliche Ursachen werden unter anderem in der globalen und damit eher grobgranularen Betrachtung von VitaminL-Sitzungen vermutet. Dennoch lassen sich bei näherer Betrachtung aller Ergebnise interessante Ansatzpunkte mit Potential zur Verbesserung der Rollenanalyse identifizieren. Diese Ansätze reichen von einer Kategorisierung der Rollenausprägungen über eine Gruppierung von CLS++-Codes, verbunden mit dem Übergang von einer einfachen linearen Korrelation zu einer multiplen Korrelation, bis hin zu einer Strukturierung von VitaminL-Sitzungen unter Berücksichtigung des VitaminL-Vorgehensmodells. Ferner ist in diesem Zusammenhang der Einsatz maschinell lernender Verfahren ebenso zu untersuchen wie die Betrachtung von nicht-linearer Regression oder die Ausweitung der Rollenanalyse auf die Inhalte von Kommunikationsbeiträgen. Diese Untersuchungen sind jedoch nicht Bestandteil der vorliegenden Arbeit, sondern in nachgelagerten Forschungen anzusiedeln.

12.2 Die Problemanalyse und ihre Ergebnisse

Im Rahmen der Studien zu Problemsituationen und deren Erkennung durch die Tutorkomponente beziehungsweise durch eine dort integrierte Problemidentifikationskomponente erfolgte zunächst eine Klassifikation der potentiell während der Zusammenarbeit innerhalb der VitaminL-IDE auftretenden Probleme. Hierbei fand eine Orientierung an den Elementen des Vorgehensmodells statt. Im Anschluss wurden Benutzertests durchgeführt, deren Ziel darin bestand, Informationen über Problemsituationen aus den Sitzungsdaten – speziell den von den Teilnehmern verwendeten CLS++-Codes – zu gewinnen. Probleme wurden bei diesen Tests von den Benutzern gemeldet und durch Beobachtungen eines menschlichen Tutors ergänzt. Zum Zweck der Informationsgewinnung wurden nach den Tests die Protokolldateien im Kontext der durch Meldung und Beobachtung gewonnenen Probleme mit intellektuellem Aufwand auf Auffälligkeiten hinsichtlich der Nutzung von CLS++-Codes analysiert. Die Ergebnisse dieser Analyse erlauben es noch nicht,

aus der Verwendung spezieller CLS++-Codes auf konkrete Problemistuationen zu schließen. Stattdessen konnten aus den Sitzungsdaten verschiedene Problemindikatoren extrahiert werden, die vermehrt in Problemsituationen zu beobachten sind. Diese Problemindikatoren sind kein sicheres Anzeichen für ein Problem, sondern besitzen eher Signalwirkung für mögliche Probleme. Insgesamt resultieren aus den Untersuchungen zu Problemsituationen – neben den Problemindikatoren – weitere Strategien und Konzepte, die bei der Entwicklung der Problemidentifikationskomponente zu berücksichtigen sind. Zu diesen Strategien gehören beispielsweise die Ausweitung des Analyseansatzes auf die Inhalte von Nachrichten (insbesondere von Kommunikationsbeiträgen), der Einsatz von maschinellen Lernverfahren in Zusammenhang mit benutzeriniitierten Problemmeldungen oder auch die Berücksichtigung des VitaminL-Vorgehensmodells, das sich ferner mit einer adaptiven Benutzeroberfläche der VitaminL-IDE kombinieren lässt.

12.3 Ausblick

Fasst man die Erkenntnisse sowohl der Rollen- als auch der Problemanalyse zusammen, so darf festgestellt werden, dass in beiden Fällen die Analyseansätze auf der Grundlage von CLS++-Codes und deren Verwendung durch die Teilnehmer von VitaminL-Sitzungen Ausgangspunkte für die jeweiligen Analysekomponenten darstellen. Dies sind jedoch aufgrund der bislang noch geringen Aussagekraft der jeweiligen Ergebnisse in der Regel um weitere Analyseverfahren zu ergänzen. In beiden Fällen sind zu diesem Zweck neben den bereits betrachteten CLS++-Codes weitere Elemente der zu von den Teilnehmern ausgelösten Nachrichtenobjekte in die Untersuchungen mit einzubeziehen. Insbesondere wird der Inhaltsanalyse von Kommunikationsbeiträgen ein hohes Potential für die Verbesserung der beiden Analysekomponenten zugesprochen.

Ferner hat sich das VitaminL-Vorgehensmodell als ein zentraler Ausgangspunkt zur möglichen Verbesserung sowohl der Rollen- als auch der Problemanalyse herausgestellt: Die Berücksichtigung des Vorgehensmodells ermöglicht eine Strukturierung der VitaminL-Sitzungen und erlaubt damit differenziertere Betrachtungen während der Zusammenarbeit. In der Folge können Analyseverfahren entwickelt, umgesetzt, in die Analysekomponenten integriert und damit in VitaminL-Sitzungen eingesetzt werden, die an die jeweils aktive Einheit des Vorgehensmodells angepasst sind. Die Umsetzung des VitaminL-Vorgehensmodells stellt letztlich auch einen Ansatz dar, das VitaminL-System um ein Werkzeug zu erweitern, mit dessen Hilfe teilnehmende Teams die Aktivitäten während ihrer Sitzungen besser aufeinander abstimmen können. Als Nebeneffekt wird die bislang noch fehlende Koordinationsunterstützung (s. Kap.8.1.1.3) realisiert, so dass die derzeit noch vorhandene Lücke zwischen Kommunikation und Kooperation (s. Kap.2.2.2; s. Abb.2.2) geschlossen werden kann.

Insgesamt stellen sowohl die Rollen- als auch die Problemanalyse wichtige Bausteine innerhalb einer Tutorkomponente dar, die virtuellen Teams während ihrer Zusammenarbeit im Kontext der Java-Programmierung angepasste Unterstüt-

zungsangebote in Problemsituationen generiert. Das aufgezeigte Weiterentwicklungspotential beider Analyseverfahren stellt eine Vielzahl von Ansätzen bereit, um die jeweilige Analysequalität beider Verfahren zu erhöhen. Zusammen mit den Ergebnissen von KÖLLE (2007) rückt die Realisierung des geplanten ICLS im Rahmen des VitaminL-Projekts in greifbare Nähe.

Anhang A

Technische Details

A.1 CLS++ – Strukturierte Zusammenarbeit

Tab. A.1: Codierung von strukturierter Kommunikation und Kooperation

Kategorie	**Element**	**Code**
Kommunikation		
A. Aufgabenplanung	Koordination	0
	Themenwechsel	1
	Zusammenfassung	2
B. Bestätigung	Anerkennen	3
	Akzeptieren	4
	Ablehnen	5
C. Moderation	Entschuldigen	6
	Zuhören,Verstehen	7
	Zustimmung einholen	8
	Tätigkeit vorschlagen	9
	Aufmerksamkeit erbitten	10
D. Anforderung	Information	11
	Einzelheiten	12
	Klärung	13
	Rechtfertigung	14
	Meinung	15
	Erklärung	16
E. Argumentation	Schlichtung	17
	Zustimmung	18
	Widerspruch	19
	Alternativangebot	20
	Folgerung	21
	Annahme	22

Fortsetzung auf nächster Seite

Tab. A.1: Codierung von strukturierter Kommunikation und Kooperation *Forts.*

Kategorie	**Element**	**Code**
	Ausnahme	23
	Zweifel	24
F. Information	Neuformulierung	25
	Führung	26
	Vorschlag	27
	Ausarbeitung	28
	Erklärung	29
	Rechtfertigung	30
	Behauptung	31
G. Motivation	Ermutigung	32
	Verstärkung	33
H. Vermittlung	Hilfe durch Tutor	34
Kooperation		
I. Dateioperationen	Erzeugen	39
	Laden	40
	Speichern	41
	Umbenennen	42
	Schliesen	43
J. Editoroperationen	Einfgen	37
	Löschen	38
	Ändern	50
	Markieren	52
K. Navigation	Navigation in einem Dokument	35
	Navigation zwischen Dokumenten	36
L. Dokumentenaustausch	Anfordern	44
	Freigeben	45
	Entziehen	46
M. Java-Operationen	Übersetzen mit Compiler	47
	Ausführen mit Interpreter	48

A.2 Das Datenmodell der Wissensbasis

Dieses Kapitel stellt detailierte Informationen über den Aufbau der Wissensbasis (s. Kap.8.4.4.3.2) bereit. Diese wurde auf einem Linux-Server mit dem Datenbanksystem *MySQL v4.1* realisiert. Es gelten folgende Vereinbarungen:

- Primärschlüssel sind in den Tabellen durch Unterstreichungen hervorgehoben.

- Bei den angegebenen Datentypen handelt es sich um die SQL-Datentypen der verwendeten MySQL-Datenbank.
- Im Anschluss an die tabellarische Darstellung des Aufbaus einer Datenbank-Tabelle folgt jeweils das zu ihrer Erzeugung gehörige SQL-Skript. Die Notation erfolgt im SQL-Dialekt des verwendeten Datenbanksystems.

Für weiterführende Informationen bezüglich MySQL sei auf `http://mysql.org/` verwiesen.

A.2.1 Die Tabelle `user`

In der Tabelle `user` sind Benutzerinformationen über alle teilnehmenden Benutzer hinterlegt. Diese Informationen werden – neben der Archivierung von Sitzungsprotokollen – auch für das Anmeldeverfahren am VitaminL-Server genutzt.

Tab. A.2: Aufbau der Wissensbasis: Tabelle `user`

Attribut	**Datentyp**	**Bedeutung**
user_id	int(10)	Jedem Eintrag dieser Tabelle ist eine eindeutige Kennung zugeordnet. Diese wird auch in anderen Tabellen verwendet, um auf Einträge in dieser Tabelle zu verweisen.
vorname	varchar(100)	Der Vorname eines Teilnehmers ist hier modelliert.
nachname	varchar(100)	Der Nachname eines Teilnehmers ist hier modelliert.
matrikel	varchar(8)	Dies ist das Attribut zur Aufnahme der Matrikelnummer eines Teilnehmers.
username	varchar(50)	Dieses Attribut modelliert die Benutzerkennung, die im System genauso eindeutig ist wie die ID eines Benutzers. Die Benutzerkennung wird – zusammen mit einem Passwort – bei Anmeldevorgängen zur Authentifizierung eines Teilnehmers gegenüber dem VitaminL-System verwendet.
farbe	int(11)	Jedem Teilnehmer wird ein RGB-Farbwert zugeordnet, der beispielsweise zur farblichen Markierung von Kommunikationsbeiträgen eines Teilnehmers verwendet wird (s. Kap.8.4.1.2.5, Abb.8.18(d)). Pro Farbanteil *Rot*,*Grün* und *Blau* wird ein 8-Bit-Wert r, g und b von 0 bis 255 verwendet, so dass sich der in der Datenbank hinterlegte Farbwert f wie folgt zusammensetzt: $f = r \cdot 256^2 + g \cdot 256^1 + b \cdot 256^0$.

Fortsetzung auf nächster Seite

Tab. A.2: Aufbau der Wissensbasis: Tabelle `user` *Forts.*

Attribut	**Datentyp**	**Bedeutung**
passwort	varchar(50)	Das Passwort eines Teilnehmers ist in diesem Attribut modelliert.
icon_dateiname	varchar(100)	Die Realisierung einfacher Awareness-Funktionen wird in der VitaminL-IDE u.a. mittels kleiner Icons umgesetzt, anhand derer jedes Teammitglied schnell identifiziert werden kann. Der Zugriff auf die zugehörigen Bildinformationen erfolgt über spezielle Icon-Dateien auf dem VitaminL-Server. Der dafür notwendige Dateiname ist in diesem Attribut modelliert.
flags	int(11)	Anhand von binären Flags werden weitere Eigenschaften von Benutzern modelliert. Derzeit sind dies folgende Flags – angegeben als hexadezimale Zahl – mit entsprechenden Bedeutungen bei gesetztem Flag: • `STATUS_FLAG_ADMIN = 0x01`: Benutzer hat Administratorrechte • `STATUS_FLAG_CHAT = 0x02`: Benutzer verwendet gewöhnlichen Chat anstelle strukturierter Kommunikation • `STATUS_FLAG_GUEST = 0x04`: Benutzer ist Gast und kann sich ohne Passwort anmelden • `STATUS_FLAG_TUTOR = 0x08`: Benutzer ist Tutor • `STATUS_FLAG_VISIBLE = 0x10`: Benutzer wird nicht angezeigt • `STATUS_FLAG_ANALYZER = 0x20`: Benutzer kann Analyse-Prototypen nutzen

Beispiel A.1: Erzeugung der Tabelle `user` (SQL-Befehl)

```
CREATE TABLE ‘user‘ (
  ‘user_id‘ int(10) unsigned NOT NULL auto_increment,
  ‘vorname‘ varchar(100) NOT NULL default ’’,
  ‘nachname‘ varchar(100) NOT NULL default ’’,
  ‘matrikel‘ varchar(8) NOT NULL default ’’,
  ‘username‘ varchar(50) NOT NULL default ’’,
  ‘passwort‘ varchar(50) NOT NULL default ’’,
  ‘farbe‘ int(11) NOT NULL default ’0’,
```

```
  ‘icon_dateiname‘ varchar(250) NOT NULL default ’’,
  ‘flags‘ int(11) NOT NULL default ’0’,
  PRIMARY KEY  (‘user_id‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 ROW_FORMAT=DYNAMIC;
```

■

A.2.2 Die Tabelle `profil`

In diese Tabelle werden die Ergebnisse aller ausgefüllten Rollenfragebögen eingespeist (s. Kap.9.2.2.1).

Tab. A.3: Aufbau der Wissensbasis: Tabelle `profil`

Attribut	Datentyp	Bedeutung
profil_id	int(10)	Dieses Attribut dient der eindeutigen Identifizierung eines Rollenprofils.
datum	datetime	Dieses Datum gibt an, wann ein Rollenprofil generiert wurde. Jedes Datum wird im Format `YYYY-MM-DD hh:mm:ss` notiert (YYYY = Jahr, MM = Monat (01-12), DD = Tag (01-31), hh = Stunde (00-23), mm = Minute (00-59), ss = Sekunde (00-59))[1].
werte	varchar(150)	Jede Antwort des 150-elementigen Fragebogens wird mit einem Zeichen `0`, `1` oder `2` codiert gemäß der in Kapitel 9.2.2.1 getroffenen Vereinbarungen.

Beispiel A.2: Erzeugung der Tabelle `profil` (SQL-Befehl)

```
CREATE TABLE ‘profil‘ (
  ‘profil_id‘ int(10) unsigned NOT NULL auto_increment,
  ‘datum‘ datetime NOT NULL default ’0000-00-00 00:00:00’,
  ‘werte‘ varchar(150) character set latin1 collate latin1_bin
                       NOT NULL default ’’,
  PRIMARY KEY  (‘profil_id‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

■

A.2.3 Die Tabelle `profil_summary`

Diese Tabelle enthält alle in der Tabelle `profil` aufgeführten Profile, wobei die Ergebnisse (Attribut `profil.werte`) – nach Rollen getrennt – verdichtet worden sind. Die Attribute `profil_summary.beichtvater` bis `profil_summary.visionaer` beinhalten somit die Rollenausprägungen nach Auswertung eines Fragebogens.

[1] Diese Spezifikations für Datumsformate gilt für alle Attribute vom SQL-Typ `datetime` aller in diesem Kapitel behandelten Tabellen der Wissensbasis.

Tab. A.4: Aufbau der Wissensbasis: Tabelle `profil_summary`

Attribut	**Datentyp**	**Bedeutung**
profil_id	int(10)	Dieses Attribut dient der eindeutigen Identifizierung eines Rollenprofils: Einträge in den Tabellen `profil_summary` und `profil`, die dieselbe ID besitzen, beziehen sich auch auf denselben Fragebogen.
beichtvater	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Beichtvater* (VitaminL: *Vertrauensperson*)
bibliothekar	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Bibliothekar* (VitaminL: *Archivar*)
trainer	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Trainer* (VitaminL: *Berater*)
arbeitstier	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Arbeitstier* (VitaminL: *Umsetzer*)
friedensstifter	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Friedensstifter* (VitaminL: *Schlichter*)
unparteiischer	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Unparteiischer* (VitaminL: *Moderator*)
herausforderer	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Herausforderer* (VitaminL: *Fragesteller*)
entdecker	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Entdecker* (VitaminL: *Informationsbeschaffer*)
pragmatiker	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Pragmatiker* (VitaminL: *Problemlöser*)
visionaer	int(11)	Ergebnisse der Rollenausprägung für die Rolle *Visionär* (VitaminL: *Planer*)

Beispiel A.3: Erzeugung der Tabelle `profil_summary` (SQL-Befehl)

```
CREATE TABLE ‘profil_summary‘ (
  ‘profil_id‘ int(10) unsigned NOT NULL default ’0’,
  ‘beichtvater‘ int(11) default ’0’,
  ‘bibliothekar‘ int(11) default ’0’,
  ‘trainer‘ int(11) default ’0’,
  ‘arbeitstier‘ int(11) default ’0’,
  ‘friedensstifter‘ int(11) default ’0’,
  ‘unparteiischer‘ int(11) default ’0’,
  ‘herausforderer‘ int(11) default ’0’,
  ‘entdecker‘ int(11) default ’0’,
  ‘pragmatiker‘ int(11) default ’0’,
  ‘visionaer‘ int(11) default ’0’,
  PRIMARY KEY  (‘profil_id‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

■

A.2.4 Die Tabelle `rel_user_profil`

Diese Tabelle stellt über die Schlüsselattribute der Tabellen `user` und `profil` (bzw. `profil_summary`) die Verknüpfung zwischen Teilnehmern und ihren Rollenprofilen her.

Tab. A.5: Aufbau der Wissensbasis: Tabelle `rel_user_profil`

Attribut	Datentyp	Bedeutung
user_id	int(10)	Die (eindeutige) Kennung eines Benutzers aus der Tabelle `user`
profil_id	int(10)	Die eindeutige Kennung eines Rollenprofils aus der Tabelle `profil` (bzw. `profil_summary`)

Beispiel A.4: Erzeugung der Tabelle `rel_user_profil` (SQL-Befehl)

```
CREATE TABLE ‘rel_user_profil‘ (
  ‘user_id‘ int(10) unsigned NOT NULL default ’0’,
  ‘profil_id‘ int(10) unsigned NOT NULL default ’0’,
  PRIMARY KEY  (‘user_id‘,‘profil_id‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

■

A.2.5 Die Tabelle `session_header`

Diese Tabelle enthält Informationen aller Sitzungen – mit Ausnahme der eigentlichen Sitzungsdaten, also derjenigen Aktionen, die die Teilnehmer einer Sitzung in deren Verlauf ausgelöst haben.

Tab. A.6: Aufbau der Wissensbasis: Tabelle `session_header`

Attribut	Datentyp	Bedeutung
session_id	int(10)	Jede Sitzung lässt sich anhand dieser Kennung eindeutig identifizieren. Diese Kennung wird auch in anderen Tabellen verwendet, um beispielsweise die zu einer bestimmten Sitzung gehörenden Benutzeraktionen zu verwalten (Tabelle `session_data`).
datum_start	datetime	Dieses Attribut spezifiziert den Anfangszeitpunkt einer Sitzung.
datum_end	datetime	Dieses Attribut spezifiziert den Endzeitpunkt einer Sitzung.
log_version	varchar(10)	Dieses Attribut enthält Informationen über den Versionsstand der diesem Sitzungseintrag zugrundeliegenden Protokolldatei.

Fortsetzung auf nächster Seite

Tab. A.6: Aufbau der Wissensbasis: Tabelle `session_header` *Forts.*

Attribut	Datentyp	Bedeutung
comment	varchar(250)	Mit diesem Attribut können zusätzliche Beschreibungen bezüglich einer Sitzung festgehalten werden. Beispiele für solche Zusatzinformationen sind *Verwendung strukturierter Kommunikation* oder *Technische Probleme nach 30 Minuten* etc..
task	varchar(250)	Dieses Attribut wird üblicherweise für Informationen über die bearbeitete Aufgabe verwendet.
groupname	varchar(250)	Der Name der an einer Sitzuung teilnehmenden Gruppe wird in diesem Attribut hinterlegt.
logname	varchar(250)	Dieses Attribut nimmt den Namen der einer Sitzung zugrundeliegenden Protokolldatei auf.

Beispiel A.5: Erzeugung der Tabelle `session_header` (SQL-Befehl)

```
CREATE TABLE ‘session_header‘ (
  ‘session_id‘ int(10) unsigned NOT NULL auto_increment,
  ‘datum_start‘ datetime NOT NULL default ’0000-00-00 00:00:00’,
  ‘datum_end‘ datetime NOT NULL default ’0000-00-00 00:00:00’,
  ‘log_version‘ varchar(10) NOT NULL default ’’,
  ‘comment‘ varchar(250) default NULL,
  ‘task‘ varchar(250) default NULL,
  ‘groupname‘ varchar(250) default NULL,
  ‘logname‘ varchar(250) default NULL,
  PRIMARY KEY  (‘session_id‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

■

A.2.6 Die Tabelle `session_data`

Diese Tabelle enthält alle Benutzeraktionen der in der Tabelle `session_header` eingetragenen Sitzungen. Alle Aktionen innerhalb einer Sitzungen werden, entsprechend ihrer (zeitlichen) Reihenfolge des Eintreffens auf dem Server, mit einer laufenden Nummer, beginnend bei 1, versehen.

Tab. A.7: Aufbau der Wissensbasis: Tabelle `session_data`

Attribut	Datentyp	Bedeutung
session_id	int(10)	Die Kennung der für einen Eintrag relevanten Sitzung
lfdnr	int(10)	Die laufende Nummer eines Eintrags

Fortsetzung auf nächster Seite

Tab. A.7: Aufbau der Wissensbasis: Tabelle `session_data` *Forts.*

Attribut	Datentyp	Bedeutung
zeit	int(10)	Der Zeitstempel eines Eintrag: Zu diesem Zeitpunkt wurde die Benutzeraktion auf dem Server erfasst und protokolliert.
sender	int(10)	Die eindeutige Absenderkennung eines Eintrags (Verweis auf Attribut `user_id` der Tabelle `user`)
code	int(10)	Dieses Attribut enthält den CLS++-Code einer Benutzeraktion (s. Tab.A.1).
content	text	Dieses Attribut enthält zusätzliche Informationen einer Benutzeraktion. Dies kann beispielsweise ein Kommunikationsbeitrag sein, ein Dokumentenname oder auch die Fehlermeldungen des Compilers.
receiver	int(10)	Falls eine Kommunikationsbotschaft (als Spezialfall einer Benutzeraktion) an einen dedizierten Empfänger gerichtet ist, so enthält dieses Attribut die Benutzerkennung dieses Empfängers (Verweis auf Attribut `user_id` der Tabelle `user`).

Beispiel A.6: Erzeugung der Tabelle `session_data` (SQL-Befehl)

```
CREATE TABLE ‘session_data‘ (
  ‘session_id‘ int(10) unsigned NOT NULL default ’0’,
  ‘lfdnr‘ int(10) unsigned NOT NULL default ’0’,
  ‘zeit‘ int(10) unsigned NOT NULL default ’0’,
  ‘sender‘ int(10) unsigned NOT NULL default ’0’,
  ‘code‘ int(10) unsigned NOT NULL default ’0’,
  ‘content‘ text,
  ‘receiver‘ int(10) unsigned default NULL,
  PRIMARY KEY  (‘session_id‘,‘lfdnr‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

■

A.2.7 Die Tabelle `rel_user_session`

Diese Tabelle stellt über die Schlüsselattribute der Tabellen `user` und `session_header` (bzw. `session_data`) die Verknüpfung zwischen Teilnehmern und ihren Sitzungsdaten her.

Tab. A.8: Aufbau der Wissensbasis: Tabelle `rel_user_session`

Attribut	Datentyp	Bedeutung
user_id	int(10)	Die (eindeutige) Kennung eines Benutzers aus der Tabelle `user`
session_id	int(10)	Die eindeutige Kennung einer Sitzung aus der Tabelle `session_header` (bzw. `session_data`)

Beispiel A.7: Erzeugung der Tabelle `rel_user_session` (SQL-Befehl)

```
CREATE TABLE ‘rel_user_session‘ (
  ‘user_id‘ int(10) unsigned NOT NULL default ’0’,
  ‘session_id‘ int(10) unsigned NOT NULL default ’0’,
  PRIMARY KEY  (‘user_id‘,‘session_id‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

■

Anhang B

Ergänzungen zu Rollenmodellen

B.1 Die Teamrollen nach Margerison und McCann

B.1.1 Die Rolle *Informierter Berater*

Informierte Berater verstehen sich auf die Beschaffung von Informationen, die sie auf leicht verständliche Weise weitervermitteln. Solche Menschen besitzen meist Geduld und Ausdauer und vertagen eine Entscheidung lieber so lange, bis sie möglichst viel über eine Aufgabe in Erfahrung gebracht haben. Außenstehende mögen dies als Entscheidungsschwäche missdeuten. Der informierte Berater hält es jedoch für besser, alles korrekt zu erfassen und nicht vorschnell einen Rat zu erteilen, der sich später als falsch erweisen könnte.

Solche Menschen spielen eine unschätzbare Rolle in der Unterstützung ihrer Teammitglieder, sie besitzen jedoch kaum starke Präferenzen im Organisations-Bereich. Sie wollen vor allem dafür sorgen, dass die Aufgabe korrekt erledigt wird und dass alle notwendigen Informationen vorliegen.

B.1.2 Die Rolle *Kreativer Innovator*

Ein kreativer Innovator ist reich an Ideen, die nicht immer mit den tatsächlichen Arbeitsverhältnissen in Einklang zu bringen sind und diese vielleicht sogar stören. Solche Menschen neigen zur Unabhängigkeit und möchten mit ihren Ideen experimentieren und sie ohne Rücksicht auf bestehende Systeme und Methoden vorantreiben. Man sollte ihnen deshalb ermöglichen, ihre Ideen zu verfolgen, ohne dass bestehende Arbeitsabläufe gestört werden, bis sich ihr neuer Ansatz bewährt hat. Viele Organisationen haben Forschungs- und Entwicklungsabteilungen (oft separat von den Produktionsabteilungen) aufgebaut, um kreativen Menschen die Chance zu bieten, ihre Ideen auf ihre Brauchbarkeit zu testen.

Jedes Team ist auf kreative Mitglieder angewiesen und muss ihnen die Möglichkeit bieten, ihre Ansichten zur Diskussion zu stellen, auch wenn sie scheinbar im Augenblick für das Tagesgeschäft eine Unterbrechung bedeuten.

B.1.3 Die Rolle *Entdeckender Promoter*

Entdeckende Promoter verstehen es meist hervorragend, eine Idee aufzugreifen und andere Leute dafür zu begeistern. Sie machen ausfindig, was innerhalb und außerhalb ihrer Organisation geschieht und vergleichen neue Ideen mit dem, was andere Leute tun. Sie können gut Kontakte knüpfen, neue Informationen und Quellen erschließen und auf diese Weise Innovationen voranbringen. Entdeckende Promoter kontrollieren vielleicht nicht immer jedes Detail, doch sie haben einen hervorragenden Blick für das große Ganze und können andere für neue Ideen begeistern. Ihnen gelingt es, Ideen zum Durchbruch zu verhelfen, auch wenn sie mit der nachfolgenden Realisierung nicht ganz so erfolgreich sind.

B.1.4 Die Rolle *Auswählender Entwickler*

Auswählende Entwickler suchen Mittel und Wege, um eine Idee in die Tat umzusetzen. Sie kümmern sich darum, ob der Markt die Innovation verlangt und prüfen sie anhand von praktischen Kriterien. Oft stellen sie einen Prototyp her oder führen Marktstudien durch. Ihr Interesse liegt in der Entwicklung einer Innovation bis zur Produktreife. Danach aber sind sie vermutlich nicht weiter an der routinemäßigen Herstellung des Produkts interessiert. Sie wenden sich lieber einem neuen Projekt zu, welches sie beurteilen und entwickeln können.

B.1.5 Die Rolle *Zielstrebiger Organisator*

Zielstrebige Organisatoren sind Menschen, die etwas vollbringen wollen. Wenn sie von einer Idee überzeugt sind, schaffen sie die Rahmenbedingungen, um sie in die Tat umzusetzen. Ihnen ist daran gelegen, klare Ziele zu setzen und dafür zu sorgen, dass alle wissen, was in ihrer jeweiligen Rolle von ihnen erwartet wird. Sie drängen darauf, dass Termine eingehalten werden. Sie können äußerst ungeduldig reagieren, sorgen aber dafür, dass Aufgaben zielkonform erfällt werden, auch wenn sie es dabei nicht allen recht machen können.

B.1.6 Die Rolle *Systematischer Umsetzer*

Systematische Umsetzer konzentrieren sich darauf, ein Produkt oder eine Dienstleistung nach einem vorgegebenen Standard herzustellen oder zu erbringen. Sie tun dies auf regelmäßiger Basis und halten ihre Aufgabe für erfüllt, wenn ihre Quoten und Pläne erreicht sind. Sie arbeiten gerne nach festgelegten Verfahren und auf systematische Weise. Die Tatsache, dass sie bereits gestern etwas produzierten,

heißt nicht, dass ihnen dies morgen langweilig erscheint. Dies steht im Gegensatz zum kreativen Innovator, der nichts davon hält, Tag für Tag eine ähnliche Arbeit zu tun, sondern bei seinen Aufgaben eine gewisse Abwechslung braucht.

Für den systematischen Umsetzer ist es wichtig, seine bestehenden Fähigkeiten einsetzen zu können und nicht immer wieder mit neuen und veränderten Arbeitsweisen konfrontiert zu werden. Ihn befriedigt es, etwas herzustellen und seine selbstgesetzten Pläne zu erfüllen.

B.1.7 Die Rolle *Kontrollierender Überwacher*

Ein kontrollierender Überwacher arbeitet gerne an detaillierten Aufgaben und sorgt dafür, dass die Zahlen und Fakten stimmen. Er arbeitet sorgfältig und genau. Seine größte Stärke ist denn auch seine Fähigkeit, sich lange Zeit auf eine spezifische Aufgabe zu konzentrieren. Dies steht im Gegensatz zum entdeckenden Promoter, der ständig verschiedene Aufgaben braucht. Der kontrollierende Überwacher hingegen verfolgt eine Aufgabe gerne gründlich und kümmert sich darum, dass die Arbeit nach Plan und exakt durchgeführt wird. Der kontrollierende Überwacher nimmt bei der Rechnungsprüfung und im Qualitätsbereich oder im Zusammenhang mit Verträgen eine äußerst wichtige Rolle ein.

B.1.8 Die Rolle *Unterstützender Stabilisator*

Unterstützende Stabilisatoren können sehr gut dafür sorgen, dass das Team eine stabile Funktionsbasis besitzt. Sie sind stolz darauf, sowohl die physische als auch die gesellschaftliche Seite der Arbeit zu unterstützen. Solche Menschen können sich zum „Gewissen“ des Teams entwickeln und viel Unterstützungsarbeit und Hilfe für Teammitglieder leisten. Sie haben meist eine ganz klare Vorstellungen davon, wie das Team geführt werden sollte. Dabei lassen sie sich von ihren Werten und Überzeugungen leiten. Wenn dem nicht entsprochen wird, können solche Menschen recht widerspenstig reagieren und ihre Interessen verteidigen. Wenn sie aber von der Aufgabe des Teams überzeugt sind, können sie zur ungeheuren Quelle der Stärke und Energie werden und ausgezeichnete Verhandlungen führen.

B.2 Das Rollenmodell nach Eunson

Eunson hat im Rahmen seiner Forschungen eine Vielzahl von Rollen identifiziert, die für ein Funktionieren der Gruppe teils erforderlich, teils nützlich und teils schädlich sind. Entsprechend dieser Differenzierung kategorisiert er diese Rollen in aufgabenorientierte, sozio-emotionale und zerstörerische Rollen. Insbesondere bei den zerstörerischen Rollen gibt er Hinweise, wie mit diesen umzugehen ist, um sie sinnvoll in den Gruppenprozess mit einzubeziehen.

B.2.1 Eunson's Aufgabenrollen

Tab. B.1: Aufgabenrollen nach Eunson

Rolle	**Beschreibung**
Initiator	Schlägt neue Wege zur Problemlösung oder Zielerreichung durch die Gruppe vor
Informationssucher	Hinterfragt neue Ideen im Hinblick auf Angemessenheit und Maßgeblichkeit
Informationsgeber	Bietet problemrelevante Tatsachen, Erfahrungen und Ideen an
Meinungssucher	Ist mehr an Werten und Glaubenssätzen als an Tatsachen interessiert
Meinungsgeber	Bietet zu vorliegenden Problemen Werte und Glaubenssätze an, keine Tatsachen
Bewerter	Entwickelt Standards der Praktikabilität, Logik, Korrektheit etc. und stellt diese auf
Ausführer	Führt Entscheidungen der Gruppe aus und befasst sich mit praktischen Details, Zeitpunkten und Methoden
Geschäftsordnungs-praktiker	Kümmert sich sowohl um äußere Notwendigkeiten (Material etc.) als auch um die Beachtung von Verfahrenserfordernissen (Termine, Vorschriften etc.)
Schriftführer	Führt Protokoll und schreibt Berichte

(Eunson, 1990, S.427f)

B.2.2 Eunson's sozio-emotionale Rollen

Tab. B.2: Sozio-emotionale Rollen nach Eunson

Rolle	**Beschreibung**
Mutmacher	Lobt und belohnt, stärkt aktiven Mitgliedern den Rücken und ermuntert die Zurückhaltenden
Friedensstifter	Vermittelt in Konfliktsituationen und versucht, Übereinstimmungen zu erzielen
Kompromissschließer	Versucht, die Harmonie in der Gruppe aufrecht zu erhalten, indem er sich selbst zurücknimmt
Spannungsmilderer	Kann das Eisbrechen und verkrampfte Sitautionen auflockern
Konfrontierer	Sieht seine Aufgabe darin, übermäßig konfliktvermeidende Situationen aufzustören; der *advocatus diaboli*

(Eunson, 1990, S.430)

B.2.3 Eunson's zerstörerische Rollen

Tab. B.3: Zerstörerische Rollen nach EUNSON (Auszug)

Rolle	**Beschreibung**
Schwätzer	Viele Worte mit wenig Inhalt kennzeichnen seine Verbalbeiträge in Sitzungen. Er liefert anderen möglicherweise wertvolle Ideen, muss aber reguliert werden, so dass auch andere zu Wort kommen können.
Detailversessener	Verliert sich gerne in unbedeutenden Kleinigkeiten. Sollte dort eingesetzt werden, wo Detailarbeit sinnvoll und notwendig ist.
Dem *„gerade was in den Sinn kommt"*	Sagt immer das Erstbeste, auch wenn es gerade nicht passt. Nützlich in Brainstormings, jedoch störend in eher orthodoxen Sitzungen
Definierer	Benötigt für alle Begriffe exakte Definitionen.
Offenhalter	Angst vor (Fehl-)Entscheidungen, kann sich nicht festlegen
Spottender Miesmacher	Muss stets seinen Kommentar abgeben, der selten durch Tatsachen gestützt oder konstruktiver Natur ist. Kann mit der Bitte um spezifische Gründe für seine Behauptungen zum Schweigen gebracht werden.
Aufschieber	Benötigt angeblich stets mehr Fakten für Entscheidungen und versucht daher, alles zu einem späteren Zeitpunkt zu verschieben.
Störer	Spricht in Diskussionen laut mit Nachbarn, so dass der Ablauf gestört wird. Nur durch strikte Anwendung von Diskussionsregeln sind Vertreter dieser Rolle in den Griff zu kriegen.
Personalisierer	Nimmt alles persönlich und reagiert entsprechend verletzt, wenn er mit anderen Meinungen und Ansichten konfrontiert wird.
Manipulierer	Ein typischer Lobbyist, der mit Lob und Schmeichelei versucht, Meinungen zu seinen Gunsten zu beeinflussen.
Dominierender	Will, dass alles nach seinem Kopf läuft, nimmt anderen damit – u.a. in Sitzungen – den für eine sinnvolle Entfaltung notwendigen Freiraum.

(EUNSON, 1990, S.430ff)

B.3 Das Rollenmodell nach Spencer & Pruss

B.3.1 Charakterisierung der Rollen nach Spencer & Pruss

Dieses Kapitel beinhaltet kurze Zusammenfassungen der in Spencer & Pruss (1995) beschriebenen Rollen. Für ausführliche Informationen zu den einzelnen Rollen sei hiermit auf (Spencer & Pruss, 1995, Kap.2) verwiesen.

B.3.1.1 Die Rolle *Visionär*

Als wesentliche Eigenschaft wird dem Visionär (s. Spencer & Pruss, 1995, S.60f) Weitsicht zugeschrieben, die es ihm ermöglicht, nicht nur den gesamten Auftrag des Teams zu überblicken, sondern diesen auch korrekt in übergeordnete Zielsetzungen (bspw. eines Unternehmens) einzubetten. Dementsprechend interessiert sich der Visionär weniger für Details, sondern vielmehr für den großen Zusammenhang, was zur Folge hat, dass er bevorzugt seine Visionen ausarbeitet. Die anfallende Arbeit muss daher von anderen Teammitgliedern erledigt werden. Der Visionär ist oftmals ein verhinderter Teamleiter und versucht in der Rolle des Visionärs seinen Eigenschaften, Fähigkeiten und Neigungen nachzukommen. Sein Führungsstil zeichnet sich eher durch Überzeugung als durch Druckausübung aus.

B.3.1.2 Die Rolle *Pragmatiker*

Als Gegenpol zum Visionär werden dem Pragmatiker (s. Spencer & Pruss, 1995, S.61ff) zwei wichtige Funktionen zugeschrieben: Zum einen zeigt der Pragmatiker dem Team Grenzen bei der Realisierung von Vorschlägen des Visionärs auf, die sich beispielsweise aus Budgetbeschränkungen ergeben, auf der anderen Seite ist der Pragmatiker konstruktiver Lieferant für realisierbare Vorschläge unter Berücksichtigung einschränkender Bedingungen. Dazu greift der Pragmatiker die Ideen des Visionärs auf, regt aber auch zu anderen Beiträgen an, um schrittweise zu einer machbaren Lösung zu gelangen. Entsprechend gilt der Pragmatiker als Realist, der als eine Art desillusionierter Visionär zu Zynismus und Skepsis neigt.

B.3.1.3 Die Rolle *Entdecker*

Der Entdecker (s. Spencer & Pruss, 1995, S.64f) versorgt das Team über externe Quellen mit denjenigen Dingen, die dem Team derzeit nicht zur Verfügung stehen, die es jedoch im Rahmen seiner Arbeit zur Zielerreichung benötigt. Dies umfasst Materialien genauso wie Unterstützung durch andere oder Informationen. Darüber hinaus ist der Entdecker auch der Botschafter des Teams, der dieses nach außen hin vertritt und repräsentiert. Damit stellt der Entdecker die Schnittstelle des Teams zu dessen Umwelt dar. Er ist daher in der Regel sehr kommunikativ und gesellig; diese Eigenschaften erleichtern ihm die Bildung von Beziehungen außerhalb des

Teams. Ferner gilt der Entdecker als neugierig und ehrgeizig, er ist von sich aus motiviert, die Welt außerhalb des Teams zu erforschen.

B.3.1.4 Die Rolle *Herausforderer*

Der Herausforderer (s. Spencer & Pruss, 1995, S.65f) neigt dazu, Bestehendes in Frage zu stellen. Dazu können Vorschläge einzelner Teammitglieder genauso gehören wie Zielsetzungen und Termine, aber auch der gesamte TEamauftrag. In einer negativen Ausprägung fungiert der Herausforderer als Pedant oder Blockierer, die positive Form des Herausforderers sorgt dafür, dass Teamziele ohne Verzögerung weiter verfolgt werden. Der Herausforder gilt als Zyniker, besitzt aber – im Gegensatz zum eher skeptischen Pragmatiker – eine optimistische Grundhaltung.

B.3.1.5 Die Rolle *Unparteiischer*

Der Unparteiische (s. Spencer & Pruss, 1995, S.66f) ist ein weitestgehend unabhängiges Teammitglied, das üblicherweise von außen hinzugezogen wird, um – ähnlich dem Herausforderer – durch Infragestellen bestehender Sachverhalte das Team neu zu beleben. Während der Herausforderer jedoch stets ein Teammitglied (und damit beispielsweise auch Firmenangehöriger) ist, ist der Unparteiische immer ein neutraler Außenstehender, der als Berater zum Team hinzugefügt wird. Entsprechend seiner Neutralität kann der Unparteiische in Konflikt- oder Dissenssituationen beide Sichtweisen nachvollziehen, was durch ein gewisses Maß an Flexibilität erleichtert wird. Darüber hinaus gilt der Unparteiische als optimistisch und entschlussfreudig.

B.3.1.6 Die Rolle *Friedensstifter*

Der Friedensstifter (s. Spencer & Pruss, 1995, S.68) sieht seine Hauptaufgabe darin, Konflikte zu schlichten beziehungsweise potentielle Konfliktherde im Vorfeld zu entdecken und daraus resultierende Problemsituationen durch geeignete präventive Maßnahmen abzuwenden. Infolgedessen besitzt der Friedensstifter ausgeprägte kommunikative Fähigkeiten und gilt eher als personenorientiert, weniger aufgabenorientiert. Dabei bleibt er meistens objektiv und tritt selbstbewusst auf.

B.3.1.7 Die Rolle *Arbeitstier*

Das Arbeitstier (s. Spencer & Pruss, 1995, S.69) bezeichnet diejenige Person im Team, die die anfallende Arbeit erledigt. Das Arbeitstier ist eher aufgaben- als personenorientiert und geht bei der Bearbeitung der eigenen Arbeiten eher unkreativ vor. Durch das Arbeiten im Team erlangt das Arbeitstier die für sich notwendige Anerkennung.

B.3.1.8 Die Rolle *Trainer*

Der Trainer (s. Spencer & Pruss, 1995, S.70f) sieht seine Aufgabe darin, das Team voranzubringen und in Krisensituationen zu motivieren. Er gilt als treibende Kraft im Team und ist in der Regel ein älteres Teammitglied, das eine gewisse Reife verbunden mit entsprechendem Erfahrungshorizont besitzt. Seine Motivation zueht der Trainer aus dem Wunsch, zusammen mit dem Team als Sieger darzustehen.

B.3.1.9 Die Rolle *Bibliothekar*

Der Bibliothekar (s. Spencer & Pruss, 1995, S.71f) ist das Archiv und damit auch interne Informationsquelle des Teams. Er zeichnet sämtliche Aktivitäten, Entscheidungen, Beurteilungen etc. auf, die vom Team durchgeführt oder beschlossen wurden, und liefert diese Informationen auf Anfrage. Dazu gehören auch die vom Entdecker beschafften Informationen. Zur Durchführung dieser Tätigkeit bedarf es einer nicht unerheblichen Menge an Fleiß und Gewissenhaftigkeit, ergänzt durch gut entwickelte Fähigkeiten hinsichtlich der Präsentation von Fakten.

B.3.1.10 Die Rolle *Beichtvater*

Der Beichtvater (s. Spencer & Pruss, 1995, S.72ff) ist das Teammitglied, mit dem die anderen Mitglieder ihre Probleme bereden können. Dies geschieht in dem sicheren Wissen, dass der Beichtvater diese auch für sich behält und vertraulich behandelt. Ehrlichkeit zählt neben Diskretion daher zu den grundlegenden Eigenschaften des Beichtvaters.

B.3.2 Der Teamfragebogen nach Spencer & Pruss

Mit einem 150 Aussagen starken Fragebogen ermitteln Spencer & Pruss (1995) Ausprägungen hinsichtlich ihres Rollenmodells, d.h. es wird versucht zu ermitteln, wie viel (oder wie wenig) eine Person zu den im Rollenmodell beschriebenen Klassifikationen tendiert. Die einzelnen Aussagen sind jeweils durch Ankreuzen von *1* (*„Da stimme ich überhaupt nicht überein"* oder *„Das trifft auf mich eigentlich nicht zu"*), *2* (weder völlige Zustimmung noch völlige Ablehnung) oder *3* (*„Da stimme ich voll zu"* oder *„Das trifft genau auf mich zu"*) zu beantworten bzw. zu bewerten.

B.3.2.1 Die Elemente des Teamfragebogens

1. Ich komme gut mit anderen Leuten aus.
2. Wenn Details übersehen werden, ist dies unwirtschaftlich und gefährlich.
3. Wenn man andere ausbildet, investiert man in die Zukunft.
4. Menschen können ersetzt werden; sie sind keine Mangelware.

5. Alle Probleme lassen sich so unterteilen, dass sie zu bewältigen sind.
6. Ich glaube, dass man manchmal etwas alleine in die Hand nehmen muss.
7. Ich glaube, dass man Details gut beachten muss; sie dürfen nicht ignoriert werden.
8. Ich brauche niemanden, der mich motiviert.
9. Viele Leute machen Vorschläge, die undurchführbar oder wertlos sind.
10. Ich sehe den Gesamtzusammenhang; Details interessieren mich nicht.
11. Unternehmen sind sehr unpersönlich geworden.
12. Wenn ich alles aufzeichne, können andere aus meinen Fehlern lernen.
13. Früher habe ich ein hektisches Leben geführt; jetzt will ich es ruhiger angehen.
14. Ich bin auch außerhalb der Arbeit aktiv (z.B. Gärtnern, Sport usw.).
15. Ich kann mit Leuten zusammenarbeiten, die unterschiedlich viel Erfahrungen und Befugnisse haben.
16. Menschen sind wichtiger als Dinge.
17. Ich habe normalerweise ein gutes Urteilsvermögen.
18. Das meiste Nützliche, das ich gelernt habe, habe ich der Schule des Lebens gelernt.
19. Ich übernehme die Führung, wenn es sein muss.
20. Ich sehe die Notwendigkeit, Beziehungen zu knüpfen, die von anderen unterschätzt werden.
21. Meistens langweilen mich lange Gespräche mit anderen.
22. Je mehr Informationen man hat, desto besser ist man gerüstet.
23. Ich glaube nicht, dass die Leute mich sympathisch finden müssen, um das zu tun, was ich sage.
24. Ich erwarte, für gute Arbeit gelobt zu werden.
25. Ich glaube, dass Klatsch und Tratsch Unruhe stiften.
26. Ich mache die Dinge auf meine Art.
27. Ich habe schon mehr als einmal Misserfolg gehabt, aber viel daraus gelernt.
28. Ich mache alles gern, was im Freien stattfindet.

29. Ich lehne Ideen, die undurchführbar scheinen, nicht gleich ab, sondern schaue sie mir genau an.
30. Ich bin eigentlich ein Optimist.
31. Man muss seine Arbeit gern machen, wenn man sie gut machen will.
32. Den Leuten fällt es schwer, mit mir über persönliche Dinge zu sprechen.
33. Auslandsreisen erweitern den Horizont.
34. Wenn ich mir ein Ziel gesetzt habe, strenge ich mich an, es auch zu erreichen.
35. Die Menschen sind das wichtigste Kapital einer Firma.
36. Ich versuche, im Team Einigkeit zu erreichen.
37. Ich möchte viele Alternativen haben, bevor ich eine Entscheidung treffe.
38. Ich glaube nicht, dass es darauf ankommt, was man weiß, sondern wen man kennt.
39. Ich bin vorsichtig.
40. Ich bin ungeduldig.
41. Wer sich über etwas Sorgen macht, kann keine gute Arbeit leisten.
42. Ich bin in Gesellschaft anderer eher ruhig und höre lieber zu.
43. Ich fühle mich gut als Teil eines Teams.
44. Neue Ideen sind oft unbrauchbar.
45. Ich glaube, dass das Team für meine Arbeit wichtig ist.
46. Mir ist Fröhlichkeit lieber als zu viel Ernst.
47. Ich glaube, dass ein breites Wissen wichtiger ist als ein tiefes Wissen.
48. Ich halte mich bei Innovationen auf dem laufenden.
49. Das Team ist wichtiger als die einzelnen Mitglieder.
50. Ich beschäftige mich gern mit komplizierten Rätseln.
51. Ich glaube, dass man mit Aufrichtigkeit am weitesten kommt.
52. Es dauert oft ein bisschen, bis ich eine Entscheidung treffe, aber dann ist es normalerweise die richtige.
53. Ehrgeiz und Wettbewerbsdenken sind allein von dem Wunsch geprägt, Sieger zu sein.

54. Wenn mir der Auftrag als Ganzes dargelegt wird, kann ich die Vorgehensweise Schritt für Schritt vorausplanen.

55. Ich streite mich selten mit anderen.

56. Ich glaube, dass jedes Problem einzeln angepackt werden muss.

57. Ich stelle meine eigenen Regeln auf.

58. Ich begeistere andere mit meinen Ideen.

59. Ich glaube, dass die Leute am besten arbeiten, wenn man ihnen klare Anweisungen und Befehle gibt.

60. Andere finden, dass ich manchmal geistesabwesend bin.

61. Ich genieße es, einen ruhigen Abend zu Hause oder mit Freunden zu verbringen.

62. Die Zukunftsvision des Teams ist für den Betrieb wichtig.

63. Die allgemeine Zukunftsvision des Teams ist äußerst wichtig.

64. Ich überlasse unausgegorenes Daherdenken anderen.

65. An allen Ideen ist etwas dran, nur müssen sie vielleicht noch überarbeitet werden.

66. Manchmal muss man einfach eine Entscheidung treffen, auch wenn noch nicht alle Fakten bekannt sind.

67. Ich glaube, dass man, wenn man eine Veränderung der Situation will, zuerst selbst etwas anders machen muss.

68. Ich habe Kontakt zu vielen Fachleuten, an die ich mich wenden kann.

69. Ich erkenne die Komplexität von Problemen.

70. Manchmal tragen meine Ideen zur Lösung von lange bestehenden oder scheinbar unlösbaren Problemen bei.

71. Schulabschlüsse und Ausbildung sind wichtig.

72. Es macht mir nichts aus, wenn man mich für ein bisschen “altmodisch” hält.

73. Ich untersuche die einzelnen Bestandteile von Problemen.

74. Regeln und Vorschriften sind zum Nutzen aller da; man sollte sich an sie halten.

75. Ich habe festgestellt, dass andere sich mir gerne anvertrauen.

76. Wenn ich eine Aufgabe zu erledigen habe, widme ich mich ihr voll und ganz.

77. Ich bin sehr neugierig und wissbegierig.

78. Mich begeistern neue Ideen.

79. Ich lasse mir Zeit, um eine Entscheidung zu treffen.

80. Ich kann aus anderen nicht das Beste herausholen, wenn sie nicht an das glauben, um das ich sie bitte.

81. Ich arbeite, um zu leben, und lebe nicht, um zu arbeiten.

82. Mit Logik an etwas heranzugehen ist der richtige Weg.

83. Ich glaube, dass man anderen gegenüber offen und ehrlich sein soll.

84. Ich arbeite gern mit anderen zusammen.

85. Ich lerne andere Leute gern näher kennen.

86. Ich finde es nicht gut, jemanden an der Nase herumzuführen.

87. Aufrichtigkeit und Ehrlichkeit im Umgang mit anderen sind wichtig.

88. Ich glaube, dass ein dichtes Netz von Beziehungen außerhalb des Teams unentbehrlich für meinen Informationsfluss ist.

89. Ich sehe Fallschlingen schon von weitem.

90. Ich habe oft originelle oder grundlegende Ideen.

91. Für mich ist meine jetzige Stellung keine Lebensstellung.

92. Dummheit ist mir ein Greuel; ich werde schnell ungeduldig mit anderen.

93. Das ganze Team muss sich über seine Zielsetzung einig sein.

94. Ich finde, dass eine ordentliche Arbeit auch eine ordentliche Bezahlung verdient.

95. Ich fühle mich nicht oft im Stress.

96. Ich finde, dass man die Leute zwingen muss, sich den Problemen zu stellen.

97. Ich glaube, dass es gut ist, wenn das Team seine Arbeit und Position von Zeit zu Zeit neu überdenkt.

98. Ich untersuche Ideen, die später vielleicht einmal von Nutzen sein werden, auch wenn sie es im Moment nicht zu sein scheinen.

99. Ich bringe das Team wieder auf den richtigen Weg zurück, wenn es davon abgewichen ist.

100. Mir gefallen Übungen zu „spielerischem Denken" und zu unorthodoxen Denkmethoden.

101. Eine freundliche Atmosphäre ist wichtig für das reibungslose Funktionieren eines Teams.

102. Ich nehme am liebsten den geraden Weg.

103. Ich habe gerne Arbeit, die mich fordert.

104. Ich mag keine Spannungen und Auseinandersetzungen.

105. Diejenigen, die am lautesten brüllen, haben nicht unbedingt die richtigen Antworten.

106. Ich stimme bei Entscheidungen nicht immer mit dem Team überein.

107. Ich glaube, dass das Team regelmäßig seine Ziele neu formulieren muss.

108. Ich habe das Gefühl, dass ich noch nicht all das erreicht habe, was ich im Lauf meiner Karriere einmal erreichen kann.

109. Ich mag es, bei Gruppenentscheidungen der Ausschlaggebende zu sein.

110. Ich glaube, dass man seine geistige Unabhängigkeit bewahren muss.

111. Ich enttäusche das Vertrauen anderer nicht.

112. Die Sekretärin spielt bei Teamsitzungen eine wichtige Rolle.

113. Ich entdecke Fehler, die anderen entgangen sind.

114. Ich glaube, dass man immer systematisch vorgehen sollte.

115. Man lernt am meisten, indem man zuhört.

116. Nichts ist „in Stein gemeißelt".

117. Ich bin von Natur aus kritisch.

118. Ich habe einen großen Freundeskreis.

119. Ich halte mit meiner Meinung bei Teamsitzungen nicht hinter dem Berg.

120. Ich finde, dass man offen miteinander sein sollte.

121. Wer laut redet, hat normalerweise Angst vor etwas.

122. Es fällt mir nicht leicht, mit anderen gut auszukommen.

123. Manchmal verlieren Leute ihren Enthusiasmus; wenn man diesen reaktivieren kann, gewinnen sie ihre Tatkraft zurück.

124. Wenn ich keine eindeutige Aufgabenstellung erhalte, weiss ich nich, wie ich die Aufgabe erledigen soll.

125. Ich finde, was gesagt werden muss, muss gesagt werden; man muss den Tatsachen ins Auge sehen.

126. Ich kann mich begeistern.

127. Ich finde Argumente, die jeden Vorschlag vernichten.

128. Ich spiele gern mit neuen Technologien herum.

129. Mir fallen oft gute Alternativen ein.

130. Meine Ideen sind nicht immer anwendbar.

131. Die meisten Menschen meinen es gut.

132. Ich lasse nicht zu, dass meine Gefühle meine Arbeit beeinflussen.

133. Es ist gut, in vielen Jobs Erfahrungen gesammelt zu haben; man verfügt dadurch über ein breitgefächertes Wissen.

134. Ich lege Wert darauf, jede Arbeit ordnungsgemäß abzuschließen.

135. Ich bleibe bei meiner Meinung, wenn ich glaube, dass ich recht habe.

136. An jedem Standpunkt ist etwas Bedenkenswertes.

137. Ich lasse nicht zu, dass meine Emotionen meine Entscheidungen beeinflussen.

138. Ich habe ein dichtes Netz von Kontakten.

139. Ich habe eine starke Persönlichkeit.

140. Sehr oft stelle ich fest, dass eine Gruppe mich in meinem Handeln hemmt; sie kann meine Ideen nicht immer in die Tat umsetzen.

141. Ich nehme nie Arbeit mit nach Hause.

142. Ohne Regeln läuft nichts.

143. Ich hasse Langeweile.

144. Schwierigkeiten sind Herausforderungen, denen man sich stellen muss.

145. Ich finde, dass jeder ein Recht auf seine Meinung hat.

146. Für jedes Problem gibt es eine Lösung.

147. Ich untersuche Ideen in ihren Einzelheiten.

148. Mir gefallen Wettkampfsportarten.

149. Es ist mir egal, ob ich in der Gruppe beliebt bin oder nicht.

150. Ich sehe Probleme, bevor sie anderen auffallen.

B.3.2.2 Hinweise zur Auswertung

Die den einzelnen Fragen zugeordneten Punkte sind anschließend gemäß der nachfolgenden Tabelle B.4 zu summieren: Eine Zahl in einem Feld der folgenden Tabelle bedeutet, dass die zugehörige Aussage (s.o.) der entsprechenden Rolle (=Spalte) zuzuordnen ist. Da jeder der zehn Rollen genau 15 Aussagen mit drei Antwortmöglichkeiten (1, 2 oder 3 Punkte) zugeordnet sind, liegen alle Ergebnisse (= Rollenausprägungen) jeweils zwischen 15 und 45 Punkten. Die Rollen mit den höchsten Punktzahlen liefern in der anschließenden Analyse wichtige Hinweise auf die von dem Befragten bevorzugten Teamfunktionen. Details dazu sind in (Spencer & Pruss, 1995, S.91ff) zu finden.

Tab. B.4: Auswertung des Teamfragebogens von Spencer & Pruss

Beichtvater	**Bibliothekar**	**Trainer**	**Arbeitstier**	**Friedensstifter**	**Unparteiischer**	**Herausforderer**	**Entdecker**	**Beichtvater**	**Visionär**
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

(Spencer & Pruss, 1995, S.93)

B.4 Das VitaminL-Rollenmodell

B.4.1 Bezug zum Rollenmodell nach Spencer & Pruss

Das VitaminL-Rollenmodell ergibt sich durch Adaption des Rollenmodells nach Spencer und Pruss (s. Kap.9.2). Die nachfolgende Tabelle B.5 fasst die Abbildung

der Rollen nach Spencer und Pruss auf die VitaminL-Rollen zusammen.

Tab. B.5: Abbildung der Rollen nach Spencer und Pruss auf VitaminL-Rollen

Rolle nach Spencer und Pruss	**Rolle des VitaminL-Modells**
Bibliothekar	Archivar
Entdecker	Informationsbeschaffer
Pragmatiker	Problemlöser
Arbeitstier	Umsetzer
Trainer	Berater
Herausforderer	Fragesteller
Visionär	Planer
Unparteiischer	Moderator
Friedensstifter	Schlichter
Beichtvater	Vertrauensperson

B.4.2 Bewertung der Rollen durch Teilnehmer

Im Dezember 2005 wurde unter den Teilnehmern einer Reihe von Benutzertests eine Umfrage durchgeführt, in welcher die Teilnehmer die Bedeutung der einzelnen VitaminL-Rollen für die synchrone Zusammenarbeit mittels der VitaminL-Software einschätzen sollten. 26 von 34 Teilnehmern stellten ihre Einschätzungen zur Auswertung zur Verfügung. Die folgende Tabelle B.6 gibt die Einzelergebnisse dieser Einschätzung wieder (s. Kap.9.3.3.2).

Tab. B.6: Einschätzung der VitaminL-Rollen durch Teilnehmer

Nr.	**Teilnehmer**	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
1	T_{170}	4	3	2	3	2	4	2	6	3	5
2	T_{185}	2	2	3	2	1	4	3	4	1	2
3	T_{286}	6	1	1	1	1	1	2	3	3	4
4	T_{287}	6	2	1	1	6	3	2	1	5	4
5	T_{288}	5	3	1	3	4	4	2	2	1	5

Fortsetzung auf nächster Seite

Tab. B.6: Einschätzung der VitaminL-Rollen durch Teilnehmer *Forts.*

Nr.	**Teilnehmer**	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
6	T_{290}	3	3	2	1	3	4		4	3	4
7	T_{293}	4	1	2	3	2	2	2	2	4	5
8	T_{294}	2	2	2	2	3	3	2	3	1	4
9	T_{296}	3	2	2	3	2	2	2	4	5	6
10	T_{297}	4	2	1	2		2	1	4	3	4
11	T_{298}		2	2	2		2		3	2	6
12	T_{299}	1	1	1	6	1	2	1	1		1
13	T_{300}	3	1		2	1	2	1	2	2	2
14	T_{302}	2	2		1	2	1	1	1	6	
15	T_{303}			3				4			
16	T_{304}	5	3	2	2	2	3	2	2	4	5
17	T_{305}	2	2	1	6	5	4	3	6	2	4
18	T_{309}	3	1	1	3	3	2	1	2	2	6
29	T_{310}	6	5	3	2	2	5	1	6	2	6
20	T_{311}	4	4	3	2	3	3	2	5	3	4
21	T_{312}		2	2	2	2	2	2	3	3	4
22	T_{313}	3	1	1	3	2	3	1	4	5	5
23	T_{314}	5	4	3	2	4	3	2	5	3	5
24	T_{316}	3	3	1		4	2	3	1	4	4
25	T_{318}	3	3	3	3	2	3	3	2	3	4
26	T_{319}	1	2	1	2	2	2	4	3	1	3
$N = 26$	ϕ_{ari} **3,5**	**2,3**	**1,8**	**2,5**	**2,6**	**2,7**	**2,0**	**3,2**	**3,0**	**4,3**	

Anhang C

Rollenprofile der VitaminL-Benutzer

C.1 Gesamtheit der erfassten Rollenprofile

C.1.1 Ergebnisse der ausgefüllten Online-Fragebögen

Tabelle gibt C.1 sämtliche 177 Rollenprofile der Teilnehmer von Benuztertests in kumulierter Form wieder, d.h. die einzelnen Antworten der Teilnehmer wurden gemäß der in Kapitel B.3.2.2 erläuterten Hinweise zur Auswertung aufsummiert. Dabei wurde das bereits in Kapitel 9.2.2.2 erläuterte korrigierte Bewertungsschema zugrundegelegt, nach welchem die Aussagen mit 0 bis 2 Punkten bewertet werden (statt mit 1 bis 3 Punkten), so dass sich alle Ergebnisse in dem Intervall $[0 \ldots 30]$ befinden. Ein Eintrag in einem Tabellenfeld sagt aus, wieviele Punkte in Summe der entsprechende Teilnehmer (=Zeile) mit den zur jeweiligen Rolle (=Spalte) gehörenden Aussagen erzielt hat. Am Ende der Tabelle finden sich darüber hinaus einige statistische Angaben zu allen Rollen wie mittlerer, minimaler und maximaler Ausprägungswert.

Tab. C.1: Kumulierte Rollenprofile der Benutzertest-Teilnehmer ($N = 177$)

Nr.	**Teilnehmer**	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
1	T_4	13	19	21	20	23	22	18	20	19	18

Fortsetzung auf nächster Seite

Tab. C.1: Kumulierte Rollenprofile der Benutzertest-Teilnehmer ($N = 177$) *Forts.*

Nr.	Teilnehmer	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
2	T_5	16	20	22	21	21	16	15	21	22	19
3	T_6	15	15	10	16	21	17	18	23	21	15
4	T_7	20	21	19	20	22	24	16	25	26	19
5	T_8	16	25	20	22	22	20	12	24	20	20
6	T_9	18	19	16	20	22	19	17	23	26	20
7	T_{11}	18	21	14	17	20	20	21	25	19	18
8	T_{12}	15	17	17	18	16	18	15	19	20	21
9	T_{13}	22	15	18	20	21	16	21	21	22	18
10	T_{14}	16	21	16	20	25	18	15	20	24	15
11	T_{15}	23	22	22	20	24	24	18	19	26	24
12	T_{16}	13	15	17	20	24	16	22	22	21	18
13	T_{17}	20	17	18	23	23	21	17	26	22	24
14	T_{18}	23	19	18	22	22	17	16	20	23	18
15	T_{19}	16	22	17	20	24	22	20	23	26	21
16	T_{20}	14	24	24	23	24	24	21	24	23	17
17	T_{21}	17	21	18	20	22	17	17	18	24	19
18	T_{22}	20	18	19	25	26	25	21	25	23	19
19	T_{23}	17	19	17	21	24	22	22	24	26	20
20	T_{25}	15	22	18	18	22	19	18	20	22	21
21	T_{26}	16	26	19	20	19	21	23	24	26	15
22	T_{27}	19	20	21	19	22	21	16	24	28	21
23	T_{28}	17	24	25	18	26	21	26	23	24	22
24	T_{29}	17	26	24	18	25	26	25	27	22	20
25	T_{30}	17	19	19	16	23	16	15	19	18	20
26	T_{31}	17	16	18	14	19	20	20	23	14	21
27	T_{32}	13	15	16	21	19	17	17	19	21	17
28	T_{33}	19	15	14	17	17	19	18	23	18	17
29	T_{34}	15	18	19	21	23	20	16	23	20	19
30	T_{35}	18	20	17	21	21	17	19	23	24	21
31	T_{36}	21	20	21	21	22	22	18	27	24	25
32	T_{37}	15	16	15	15	18	16	14	20	21	14
33	T_{38}	19	22	18	19	23	21	19	22	19	17

Fortsetzung auf nächster Seite

Tab. C.1: Kumulierte Rollenprofile der Benutzertest-Teilnehmer ($N = 177$) *Forts.*

Nr.	Teilnehmer	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
34	T_{39}	16	21	20	16	19	23	17	25	23	19
35	T_{40}	18	17	17	23	24	19	19	21	23	22
36	T_{41}	17	21	17	16	18	20	18	22	19	21
37	T_{42}	14	17	17	19	22	15	17	23	23	17
38	T_{43}	17	14	17	18	20	21	16	22	20	19
39	T_{44}	15	18	17	20	23	22	14	19	25	22
40	T_{45}	12	15	16	21	20	17	15	23	23	19
41	T_{46}	17	19	15	16	20	18	20	22	23	19
42	T_{47}	13	17	24	14	22	19	25	20	19	17
43	T_{48}	16	27	21	21	25	19	20	22	23	19
44	T_{49}	19	17	22	16	22	24	18	22	22	18
45	T_{50}	16	19	20	19	24	21	17	23	21	16
46	T_{51}	15	18	15	22	21	16	15	22	22	21
47	T_{52}	17	25	19	25	23	23	20	25	25	24
48	T_{53}	12	20	21	17	26	21	19	26	22	23
49	T_{54}	18	19	24	20	25	22	18	22	24	21
50	T_{55}	13	10	15	21	22	12	14	25	21	21
51	T_{56}	13	22	18	19	20	22	14	21	21	18
52	T_{57}	16	18	21	13	20	20	19	24	16	22
53	T_{58}	15	15	14	15	15	15	17	15	15	15
54	T_{59}	12	21	17	20	22	21	17	23	23	21
55	T_{60}	15	18	19	18	19	20	22	26	24	20
56	T_{61}	13	24	18	17	23	26	20	27	25	20
57	T_{62}	13	18	10	21	22	19	16	22	23	19
58	T_{63}	16	20	16	18	22	20	18	27	24	20
59	T_{64}	14	24	17	17	22	21	17	25	25	21
60	T_{65}	20	19	20	23	24	21	20	24	22	23
61	T_{66}	10	7	9	6	8	4	14	6	5	3
62	T_{67}	17	20	19	17	25	20	23	22	19	18
63	T_{68}	18	16	14	22	21	20	14	21	24	19
64	T_{69}	16	22	13	23	18	16	16	19	17	19
65	T_{70}	18	18	25	17	26	22	20	23	22	19

Fortsetzung auf nächster Seite

Tab. C.1: Kumulierte Rollenprofile der Benutzertest-Teilnehmer ($N = 177$) *Forts.*

Nr.	Teilnehmer	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
66	T_{71}	15	18	19	18	22	22	17	22	21	21
67	T_{72}	17	21	17	17	26	23	14	22	25	19
68	T_{73}	19	25	18	25	26	21	15	26	24	22
69	T_{74}	13	20	15	20	19	15	20	23	22	18
70	T_{75}	13	18	19	23	19	18	17	27	25	21
71	T_{76}	21	19	18	22	22	17	19	21	22	23
72	T_{77}	18	17	16	24	22	22	14	22	22	21
73	T_{78}	10	21	18	15	19	15	16	20	25	19
74	T_{79}	15	18	19	19	23	20	18	23	21	16
75	T_{80}	20	16	15	24	24	20	14	23	21	20
76	T_{81}	20	19	14	24	22	20	20	25	20	18
77	T_{82}	13	22	22	21	21	22	21	24	25	18
78	T_{83}	17	19	21	17	26	20	19	24	22	21
79	T_{84}	11	20	16	16	19	20	17	24	25	21
80	T_{85}	18	20	17	19	25	17	17	23	23	22
81	T_{86}	19	17	14	19	22	18	16	20	22	20
82	T_{87}	13	21	17	20	23	21	17	23	24	18
83	T_{88}	20	22	21	21	25	26	19	26	25	18
84	T_{89}	15	18	19	22	22	21	16	24	22	15
85	T_{90}	16	18	17	16	14	16	11	17	19	20
86	T_{91}	16	14	19	19	22	18	19	20	21	19
87	T_{92}	16	21	18	22	23	19	23	23	23	18
88	T_{93}	18	6	8	15	17	17	16	22	20	19
89	T_{94}	17	15	16	18	20	21	22	22	21	18
90	T_{95}	13	23	20	20	25	19	21	22	23	19
91	T_{96}	14	19	17	20	21	17	22	23	19	17
92	T_{97}	22	17	18	25	27	25	15	27	27	22
93	T_{98}	17	11	10	18	16	15	10	19	27	13
94	T_{99}	20	22	27	19	20	22	22	24	24	22
95	T_{100}	12	14	8	15	16	12	13	21	25	12
96	T_{101}	16	19	21	19	22	20	17	19	19	20
97	T_{102}	16	18	18	17	25	21	20	24	23	19

Fortsetzung auf nächster Seite

Tab. C.1: Kumulierte Rollenprofile der Benutzertest-Teilnehmer ($N = 177$) *Forts.*

Nr.	Teilnehmer	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
98	T_{162}	16	20	20	17	23	23	16	28	27	22
99	T_{163}	16	13	14	23	21	18	17	23	24	20
100	T_{165}	10	21	15	19	22	19	20	26	25	21
101	T_{166}	17	19	20	19	19	18	19	23	21	20
102	T_{167}	14	17	13	19	21	18	15	23	23	19
103	T_{168}	17	20	16	18	24	23	17	21	24	22
104	T_{169}	17	17	13	23	26	22	16	21	26	15
105	T_{170}	17	17	18	20	24	17	15	20	22	20
106	T_{171}	21	21	19	21	22	23	12	24	22	23
107	T_{172}	14	22	20	18	20	22	19	22	24	19
108	T_{174}	14	19	20	14	12	21	16	21	21	22
109	T_{180}	18	19	16	21	21	21	20	20	21	21
110	T_{183}	17	26	19	20	28	24	17	23	26	25
111	T_{188}	15	24	23	22	22	22	19	20	24	19
112	T_{190}	16	20	18	19	22	23	15	22	22	22
113	T_{198}	20	17	17	23	22	23	19	18	19	19
114	T_{200}	11	19	18	20	18	18	14	23	24	19
115	T_{201}	18	13	16	22	21	23	13	19	19	20
116	T_{203}	18	21	16	19	21	21	19	25	26	23
117	T_{206}	15	22	21	25	22	21	22	27	22	26
118	T_{208}	15	21	19	21	21	19	21	26	23	18
119	T_{209}	18	18	17	20	19	17	20	24	21	23
120	T_{210}	13	22	18	17	20	20	17	19	21	18
121	T_{211}	15	18	17	20	23	22	15	21	24	21
122	T_{212}	17	23	21	23	24	26	20	25	25	23
123	T_{216}	15	22	21	22	22	24	19	25	26	22
124	T_{217}	14	16	15	16	19	19	15	21	21	19
125	T_{219}	11	11	13	15	22	16	12	15	17	16
126	T_{220}	13	21	19	17	18	18	17	21	21	16
127	T_{230}	12	23	15	13	23	24	19	23	25	21
128	T_{233}	18	17	18	19	21	22	18	19	25	16
129	T_{249}	22	17	19	19	19	16	19	26	18	18

Fortsetzung auf nächster Seite

Tab. C.1: Kumulierte Rollenprofile der Benutzertest-Teilnehmer ($N = 177$) *Forts.*

Nr.	Teilnehmer	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
130	T_{250}	18	20	18	21	25	25	19	23	20	19
131	T_{251}	14	22	20	18	19	19	21	22	25	19
132	T_{252}	11	20	16	19	22	20	15	25	22	20
133	T_{254}	17	14	15	18	22	18	15	19	20	20
134	T_{255}	16	12	9	23	17	21	19	21	19	20
135	T_{256}	15	21	21	17	23	19	24	25	22	19
136	T_{257}	19	18	13	15	24	17	20	23	24	20
137	T_{260}	14	13	11	17	19	19	11	19	21	13
138	T_{261}	22	27	22	20	25	19	17	24	22	18
139	T_{262}	19	13	17	20	17	18	14	18	21	20
140	T_{264}	13	21	19	21	25	19	20	25	28	19
141	T_{267}	20	13	15	21	24	23	19	23	17	20
142	T_{271}	14	16	17	22	19	18	16	22	24	19
143	T_{273}	19	22	16	23	19	22	22	21	17	18
144	T_{277}	19	24	20	23	24	25	17	25	25	20
145	T_{278}	16	17	21	17	25	26	21	28	30	21
146	T_{280}	15	15	15	15	15	15	15	15	15	15
147	T_{280}	15	15	15	15	15	15	15	15	15	15
148	T_{281}	13	17	18	17	21	18	19	23	24	18
149	T_{282}	22	17	18	20	21	23	17	23	24	23
150	T_{283}	18	17	13	15	16	23	13	21	20	18
151	T_{285}	21	22	18	22	25	17	24	19	25	19
152	T_{286}	15	9	16	22	19	15	15	23	19	20
153	T_{287}	24	20	20	22	25	24	23	22	20	22
154	T_{288}	12	24	18	18	20	22	19	25	27	17
155	T_{290}	16	17	18	20	22	18	22	19	20	18
156	T_{291}	16	22	20	24	25	23	13	27	22	20
157	T_{292}	17	18	14	19	23	18	18	21	23	19
158	T_{293}	16	16	12	18	20	16	14	18	23	22
159	T_{294}	16	23	18	20	23	19	17	25	25	23
160	T_{296}	14	19	15	14	21	19	13	22	21	18
161	T_{297}	16	23	18	23	24	16	17	22	26	20

Fortsetzung auf nächster Seite

Tab. C.1: Kumulierte Rollenprofile der Benutzertest-Teilnehmer ($N = 177$) *Forts.*

Nr.	Teilnehmer	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
162	T_{298}	21	20	26	23	24	27	21	25	25	23
163	T_{299}	18	12	11	16	17	20	16	23	17	24
164	T_{300}	14	16	17	16	18	15	16	19	20	18
165	T_{302}	17	15	20	23	24	17	17	20	21	24
166	T_{303}	15	20	23	17	18	23	22	17	17	15
167	T_{304}	18	12	13	13	17	16	16	20	21	15
168	T_{305}	18	11	12	22	20	18	16	21	18	15
169	T_{308}	15	18	17	17	20	17	16	21	24	21
170	T_{309}	15	14	19	20	24	22	19	18	24	21
171	T_{310}	16	21	23	18	25	19	24	20	23	24
172	T_{311}	16	17	17	23	24	21	19	24	23	23
173	T_{312}	19	21	17	19	21	20	17	19	21	15
174	T_{315}	18	19	18	19	19	22	13	22	23	20
175	T_{316}	19	16	17	21	21	20	17	21	23	19
176	T_{318}	21	22	22	22	23	21	17	23	26	21
177	T_{319}	18	20	18	25	21	20	19	24	27	21

C.1.2 Statistische Betrachtungen

Tabelle C.2 enthält einige statistische Angaben zu allen Rollen wie minimaler und maximaler Ausprägungswert sowie arithmetisches und geometrisches Mittel. Als Grundlage dienen alle 177 erfassten Rollenprofile (s. Tab.C.1).

Tab. C.2: Statistische Betrachtungen aller erfassten Rollenprofile ($N = 177$)

	Archivar	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
ϕ_{ari}	16,4	18,7	17,6	19,3	21,5	19,8	17,7	22,2	22,2	19,4
ϕ_{ari}	16,1	18,3	17,3	19,1	21,3	19,5	17,4	22,0	21,9	19,1
min	10	6	8	6	8	4	10	6	5	3
max	24	27	27	25	28	27	26	28	30	26

C.1.3 Absolute Häufigkeiten der Rollenausprägungen

Aus der Tabelle C.1 lassen sich die absoluten Häufigkeiten der einzelnen Rollenausprägungen ableiten. Tabelle C.3 gibt Auskunft darüber, wieviele Teilnehmer für eine bestimmte Rolle (=Spalte) einen bestimmten Ausprägungswert (=Zeile) besitzen.

Tab. C.3: Absolute Häufigkeiten der Rollenausprägungen ($N = 177$)

Ausprägung	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
3	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0
6	0	1	0	1	0	0	0	1	0	0
7	0	1	0	0	0	0	0	0	0	0
8	0	0	2	0	1	0	0	0	0	0
9	0	1	2	0	0	0	0	0	0	0
10	3	1	3	0	0	0	1	0	0	0
11	4	3	2	0	0	0	2	0	0	0

Fortsetzung auf nächster Seite

Tab. C.3: Absolute Häufigkeiten der Rollenausprägungen ($N = 177$) *Forts.*

Ausprägung	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
12	6	3	2	0	1	2	3	0	0	1
13	17	5	7	3	0	0	6	0	0	2
14	13	5	8	4	1	0	12	0	1	1
15	24	11	14	10	3	9	18	4	3	12
16	28	9	15	11	4	13	20	0	1	5
17	24	21	27	19	6	16	30	2	6	8
18	22	19	30	17	7	18	13	5	4	25
19	12	20	20	22	19	20	25	17	13	34
20	10	19	14	28	14	22	17	15	12	27
21	6	20	14	19	21	23	10	20	25	26
22	5	18	6	16	35	21	10	25	24	15
23	2	5	3	17	18	14	4	34	23	11
24	1	7	4	4	20	8	3	18	24	6
25	0	3	2	6	17	4	2	18	20	2
26	0	3	1	0	8	5	1	8	12	1
27	0	2	1	0	1	1	0	8	5	0
28	0	0	0	0	1	0	0	2	2	0
29	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	1	0

C.1.4 Graphische Zusammenfassung

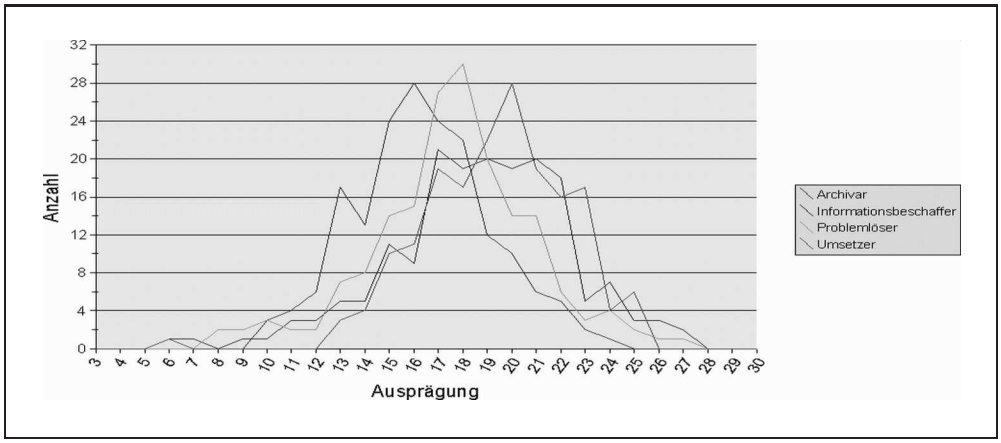

Abb. 3.1: Verteilung der Ausprägungen: Technische Rollen ($N = 177$)

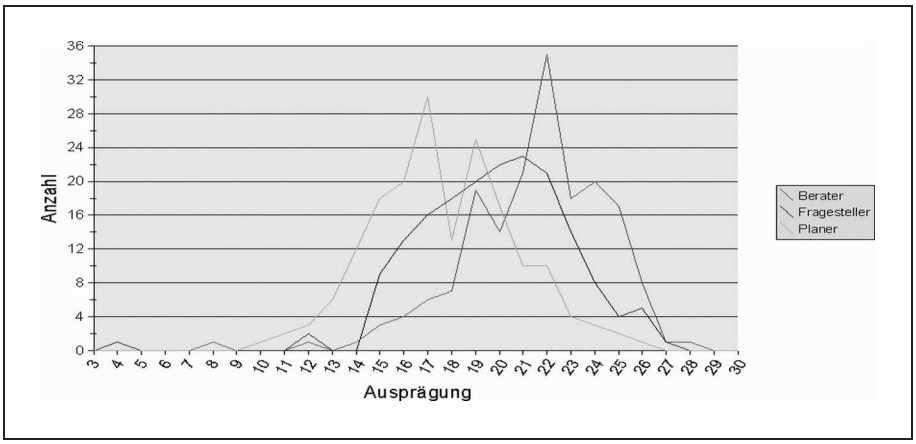

Abb. 3.2: Verteilung der Ausprägungen: Integrierende Rollen ($N = 177$)

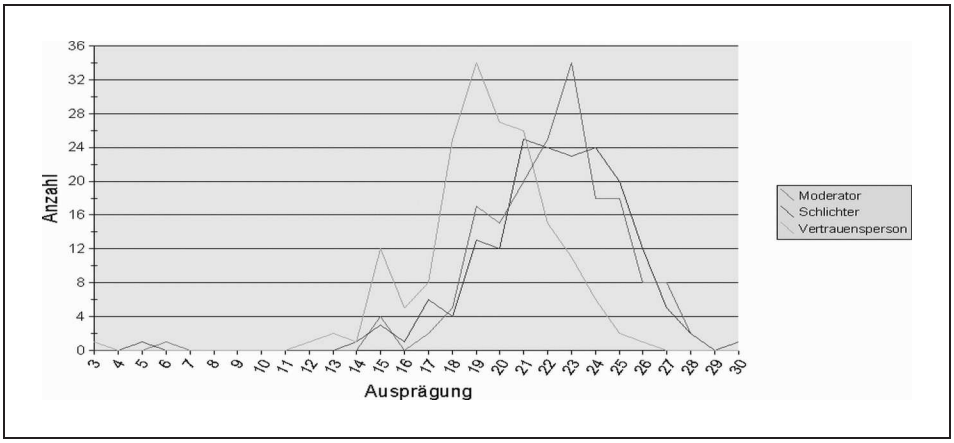

Abb. 3.3: Verteilung der Ausprägungen: Soziale Rollen ($N = 177$)

C.2 Rollenprofile der Rollenanalyse

Für die Rollenanalyse wurden zwischen Dezember 2004 und März 2005 die Rollenprofile von 29 Teilnehmern erfasst und zur Auswertung herangezogen.

C.2.1 Ergebnisse der ausgefüllten Online-Fragebögen

Die Ergebnisse der Befragung sind in Tabelle C.4 zusammengefasst.

Tab. C.4: Rollenprofile als Grundlage der Rollenanalyse ($N = 29$)

Nr.	Teilnehmer	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
1	T_4	13	19	21	20	23	22	18	20	19	18
2	T_5	16	20	22	21	21	16	15	21	22	19
3	T_6	15	15	10	16	21	17	18	23	21	15
4	T_7	20	21	19	20	22	24	16	25	26	19
5	T_8	16	25	20	22	22	20	12	24	20	20
6	T_9	18	19	16	20	22	19	17	23	26	20
7	T_{11}	18	21	14	17	20	20	21	25	19	18
8	T_{12}	15	17	17	18	16	18	15	19	20	21
9	T_{13}	22	15	18	20	21	16	21	21	22	18
10	T_{14}	16	21	16	20	25	18	15	20	24	15
11	T_{15}	23	22	22	20	24	24	18	19	26	24
12	T_{16}	13	15	17	20	24	16	22	22	21	18
13	T_{17}	20	17	18	23	23	21	17	26	22	24
14	T_{18}	23	19	18	22	22	17	16	20	23	18
15	T_{19}	16	22	17	20	24	22	20	23	26	21
16	T_{20}	14	24	24	23	24	24	21	24	23	17
17	T_{21}	17	21	18	20	22	17	17	18	24	19
18	T_{22}	20	18	19	25	26	25	21	25	23	19
19	T_{23}	17	19	17	21	24	22	22	24	26	20
20	T_{25}	15	22	18	18	22	19	18	20	22	21
21	T_{26}	16	26	19	20	19	21	23	24	26	15
22	T_{27}	19	20	21	19	22	21	16	24	28	21

Fortsetzung auf nächster Seite

Tab. C.4: Rollenprofile als Grundlage der Rollenanalyse ($N = 29$) *Forts.*

Nr.	Teilnehmer	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
23	T_{28}	17	24	25	18	26	21	26	23	24	22
24	T_{29}	17	26	24	18	25	26	25	27	22	20
25	T_{30}	17	19	19	16	23	16	15	19	18	20
26	T_{31}	17	16	18	14	19	20	20	23	14	21
27	T_{32}	13	15	16	21	19	17	17	19	21	17
28	T_{33}	19	15	14	17	17	19	18	23	18	17
29	T_{34}	15	18	19	21	23	20	16	23	20	19

C.2.2 Statistische Betrachtungen

Tabelle C.5 enthält einige statistische Angaben zu allen Rollen wie minimaler und maximaler Ausprägungswert sowie arithmetisches und geometrisches Mittel. Als Grundlage dienen die genannten 29 erfassten Rollenprofile (s. Tab.C.4).

Tab. C.5: Statistische Betrachtungen der Rollenanalyseprofile ($N = 29$)

	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
ϕ_{ari}	17,1	19,7	18,5	19,7	22,1	19,9	18,5	22,3	22,3	19,2
ϕ_{geo}	16,9	19,4	18,2	19,5	22,0	19,7	18,2	22,2	22,1	19,0
min	13	15	10	14	16	16	12	18	14	15
max	23	26	25	25	26	26	26	27	28	24

C.2.3 Absolute Häufigkeiten der Rollenausprägungen

Aus Tabelle C.4 lassen sich die absoluten Häufigkeiten der einzelnen Rollenausprägungen ableiten. Tabelle C.6 gibt Auskunft darüber, wieviele Teilnehmer für eine bestimmte Rolle (=Spalte) einen bestimmten Ausprägungswert (=Zeile) besitzen.

Tab. C.6: Absolute Häufigkeiten der Rollenausprägungen ($N = 29$)

Ausprägung	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
3	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0
6	0	1	0	1	0	0	0	1	0	0
7	0	1	0	0	0	0	0	0	0	0
8	0	0	2	0	1	0	0	0	0	0
9	0	1	2	0	0	0	0	0	0	0
10	3	1	3	0	0	0	1	0	0	0
11	4	3	2	0	0	0	2	0	0	0
12	6	3	2	0	1	2	3	0	0	1
13	17	5	7	3	0	0	6	0	0	2
14	13	5	8	4	1	0	12	0	1	1
15	24	11	14	10	3	9	18	4	3	12
16	28	9	15	11	4	13	20	0	1	5
17	24	21	27	19	6	16	30	2	6	8
18	22	19	30	17	7	18	13	5	4	25
19	12	20	20	22	19	20	25	17	13	34
20	10	19	14	28	14	22	17	15	12	27
21	6	20	14	19	21	23	10	20	25	26
22	5	18	6	16	35	21	10	25	24	15
23	2	5	3	17	18	14	4	34	23	11
24	1	7	4	4	20	8	3	18	24	6
25	0	3	2	6	17	4	2	18	20	2
26	0	3	1	0	8	5	1	8	12	1
27	0	2	1	0	1	1	0	8	5	0
28	0	0	0	0	1	0	0	2	2	0

Fortsetzung auf nächster Seite

Tab. C.6: Absolute Häufigkeiten der Rollenausprägungen ($N = 29$) *Forts.*

Ausprägung	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
29	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	1	0

C.2.4 Graphische Zusammenfassung

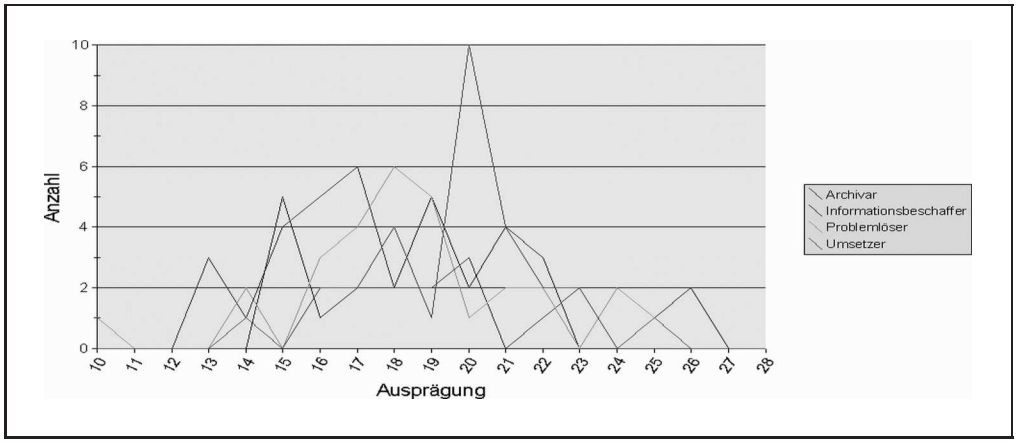

Abb. 3.4: Verteilung der Ausprägungen: Technische Rollen ($N = 29$)

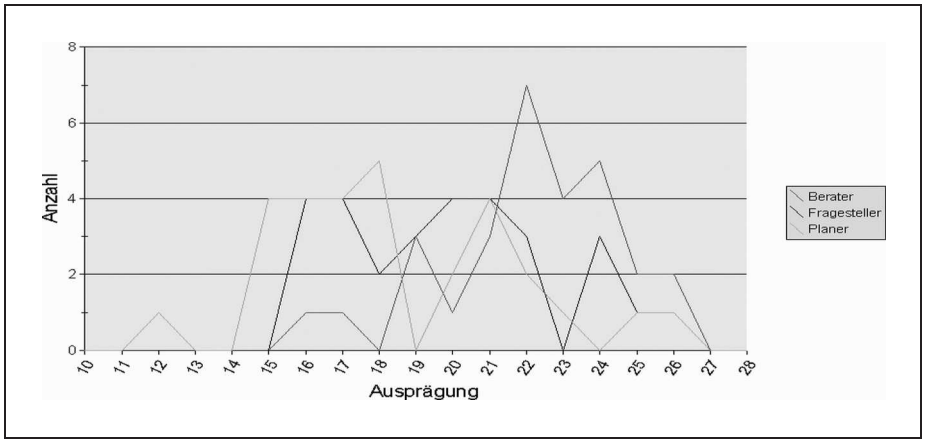

Abb. 3.5: Verteilung der Ausprägungen: Integrierende Rollen ($N = 29$)

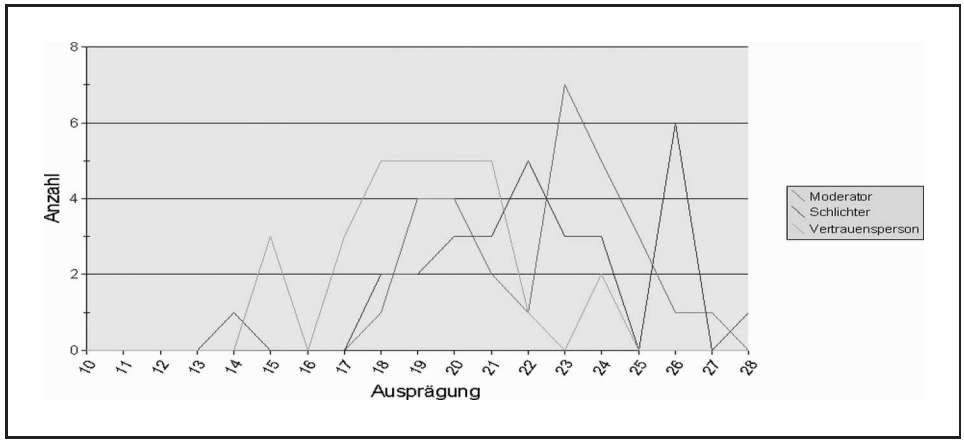

Abb. 3.6: Verteilung der Ausprägungen: Soziale Rollen ($N = 29$)

C.3 Rollenprofile der Evaluierung der Rollenanalyse

Für die Evaluierung der prototypischen Komponente zur Rollenanalyse wurden im Dezember 2005 die Rollenprofile von 24 Teilnehmern erfasst und zur Auswertung herangezogen.

C.3.1 Ergebnisse der ausgefüllten Online-Fragebögen

Die Ergebnisse der Befragung sind in Tabelle C.7 zusammengefasst.

Tab. C.7: Rollenprofile der Evaluierung der Rollenanalyse ($N = 24$)

Nr.	Teilnehmer	Archivar	Informationsbeschaffer	Problemlöser	Umsetzer	Berater	Fragesteller	Planer	Moderator	Schlichter	Vertrauensperson
1	T_{209}	23	18	19	20	21	24	17	18	17	20
2	T_{285}	19	21	25	22	25	19	17	22	18	24
3	T_{286}	20	15	19	22	19	23	15	9	16	15
4	T_{287}	22	24	25	22	20	22	24	20	20	23
5	T_{291}	20	16	25	24	22	27	23	22	20	13
6	T_{292}	19	17	23	19	23	21	18	18	14	18

Fortsetzung auf nächster Seite

Tab. C.7: Rollenprofile der Evaluierung der Rollenanalyse ($N = 24$) *Forts.*

Nr.	**Teilnehmer**	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
7	T_{293}	22	16	20	18	23	18	16	16	12	14
8	T_{294}	23	16	23	20	25	25	19	23	18	17
9	T_{296}	18	14	21	14	21	22	19	19	15	13
10	T_{297}	20	16	24	23	26	22	16	23	18	17
11	T_{298}	23	21	24	23	25	25	27	20	26	21
12	T_{299}	24	18	17	16	17	23	20	12	11	16
13	T_{300}	18	14	18	16	20	19	15	16	17	16
14	T_{302}	24	17	24	23	21	20	17	15	20	17
15	T_{303}	15	15	18	17	17	17	23	20	23	22
16	T_{304}	15	18	17	13	21	20	16	12	13	16
17	T_{305}	15	18	20	22	18	21	18	11	12	16
18	T_{308}	21	15	20	17	24	21	17	18	17	16
19	T_{309}	21	15	24	20	24	18	22	14	19	19
20	T_{310}	24	16	25	18	23	20	19	21	23	24
21	T_{311}	23	16	24	23	23	24	21	17	17	19
22	T_{312}	15	19	21	19	21	19	20	21	17	17
23	T_{315}	20	18	19	19	23	22	22	19	18	13
24	T_{316}	19	19	21	21	23	21	20	16	17	17

C.3.2 Statistische Betrachtungen

Tabelle C.8 enthält einige statistische Angaben zu allen Rollen wie minimaler und maximaler Ausprägungswert sowie arithmetisches und geometrisches Mittel. Als Grundlage dienen die genannten 24 erfassten Rollenprofile (s. Tab.C.7).

Tab. C.8: Statistische Betrachtungen der Evaluierungsprofile ($N = 24$)

	Archivar	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
ϕ_{ari}	20,1	17,2	21,5	19,6	21,9	21,4	19,2	17,6	17,4	17,6
ϕ_{geo}	19,9	17,0	21,3	19,4	21,7	21,2	19,0	17,1	17,1	17,3
min	15	14	17	13	17	17	15	9	11	13
max	24	24	25	24	26	27	27	23	26	24

C.3.3 Absolute Häufigkeiten der Rollenausprägungen

Aus der Tabelle tab-analyse-rollen-profile-2 lassen sich die absoluten Häufigkeiten der einzelnen Rollenausprägungen ableiten. Tabelle C.9 gibt Auskunft darüber, wieviele Teilnehmer für eine bestimmte Rolle (=Spalte) einen bestimmten Ausprägungswert (=Zeile) besitzen.

Tab. C.9: Absolute Häufigkeiten der Rollenausprägungen ($N = 24$)

Ausprägung	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
9	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1	1	0
12	0	0	0	0	0	0	0	2	2	0
13	0	0	0	1	0	0	0	0	1	3
14	0	2	0	1	0	0	0	1	1	1
15	4	4	0	0	0	0	2	1	1	1
16	0	6	0	2	0	0	3	3	1	5
17	0	2	2	2	2	1	4	1	6	5

Fortsetzung auf nächster Seite

Tab. C.9: Absolute Häufigkeiten der Rollenausprägungen ($N = 24$) *Forts.*

Ausprägung	**Archivar**	**Informationsbeschaffer**	**Problemlöser**	**Umsetzer**	**Berater**	**Fragesteller**	**Planer**	**Moderator**	**Schlichter**	**Vertrauensperson**
18	2	5	2	2	1	2	2	3	4	1
19	3	2	3	3	1	3	3	2	1	2
20	4	0	3	3	2	3	3	3	3	1
21	2	2	3	1	5	4	1	2	0	1
22	2	0	0	4	1	4	2	2	0	1
23	4	0	2	4	6	2	2	2	2	1
24	3	1	5	1	2	2	1	0	0	2
25	0	0	4	0	3	2	0	0	0	0
26	0	0	0	0	1	0	0	0	1	0
27	0	0	0	0	0	1	1	0	0	0

C.3.4 Graphische Zusammenfassung

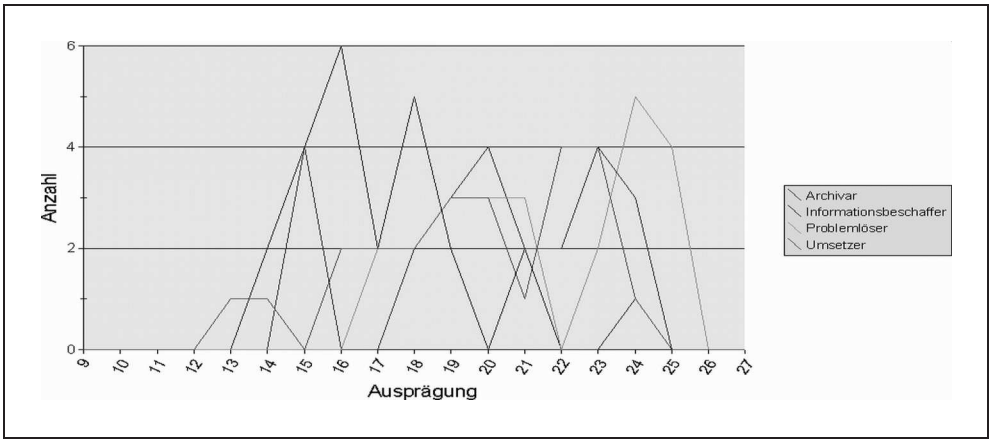

Abb. 3.7: Verteilung der Ausprägungen: Technische Rollen ($N = 24$)

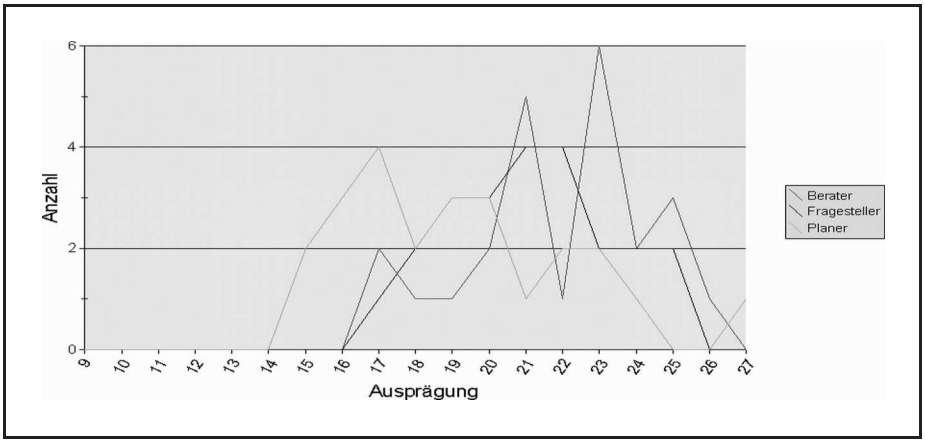

Abb. 3.8: Verteilung der Ausprägungen: Integrierende Rollen ($N = 24$)

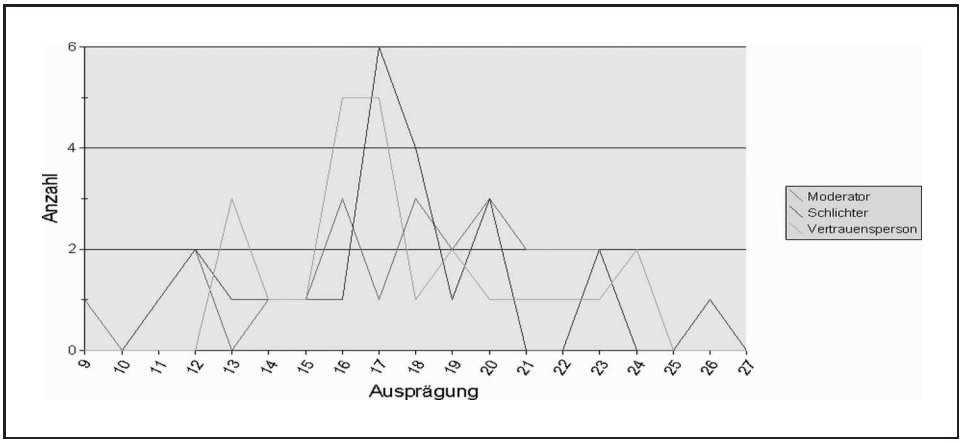

Abb. 3.9: Verteilung der Ausprägungen: Soziale Rollen ($N = 24$)

Anhang D

Tabellen zur Rollenanalyse

D.1 Kritische Werte $r_{\alpha;m}$ für den Maßkorrelationskoeffizienten r

Tab. D.1: Zufallshöchstwerte des Maßkorrelationskoeffizienten r

Freiheitsgrad m	Irrtumswahrscheinlichkeit α			
	5 %	**1 %**	0,27 %	0,1 %
	Zufallshöchstwerte von r			
5	0,75	**0,87**	0,93	0,95
10	0,58	**0,71**	0,78	0,82
15	0,48	**0,61**	0,68	0,72
20	0,42	**0,53**	0,61	0,65
25	0,38	**0,49**	0,55	0,60
30	0,35	**0,45**	0,51	0,55
35	0,32	**0,42**	0,48	0,52
40	0,30	**0,39**	0,45	0,49
50	0,27	**0,35**	0,41	0,44
60	0,25	**0,33**	0,37	0,41
70	0,23	**0,30**	0,35	0,38
80	0,22	**0,28**	0,33	0,36
90	0,21	**0,26**	0,31	0,34
100	0,19	**0,25**	0,29	0,32
120	0,18	**0,23**	0,27	0,30
150	0,16	**0,21**	0,24	0,26
200	0,14	**0,18**	0,21	0,23
300	0,11	**0,15**	0,17	0,19
400	0,10	**0,13**	0,15	0,16
500	0,09	**0,11**	0,13	0,15
700	0,07	**0,10**	0,11	0,12

Fortsetzung auf nächster Seite

Tab. D.1: Zufallshöchstwerte des Maßkorrelationskoeffizienten r *Forts.*

Freiheitsgrad m	Irrtumswahrscheinlichkeit α			
	5 %	**1 %**	0,27 %	0,1 %
	Zufallshöchstwerte von r			
900	0,06	**0,09**	0,10	0,11
1000	kleiner als	**kleiner als**	kleiner als	kleiner als
und mehr	0,06	**0,09**	0,10	0,11

(Clauss et al., 1995, S.407)

Für $\alpha = 1\%$ sind die Höchstwerte von r fett markiert. Nicht in der Tabelle aufgeführte Freiheitsgrade und zugehörige Zufallshöchstwerte für r sind durch lineare Interpolation abzuschätzen. Für Freiheitsgrade m aus dem Intervall $[5 \ldots 70]$ liefert die Funktion $f(m) = \frac{1}{\sqrt{a+bm}}$ mit $a = 0,56811$ und $b = 0,15061$ eine hinreichend genaue Näherung für die Vergleichswerte $r_{\alpha;m}$ bei $\alpha = 1\% = 0,01$.

Diese Näherungsfunktion ist vor allem dann nützlich, wenn die Auswertung einer umfangreichen Datenmenge wie der hier vorliegenden (s. Kap.10.1.1.3.2) mittels geeigneter Software in weiten Teilen automatisiert stattfinden muss.

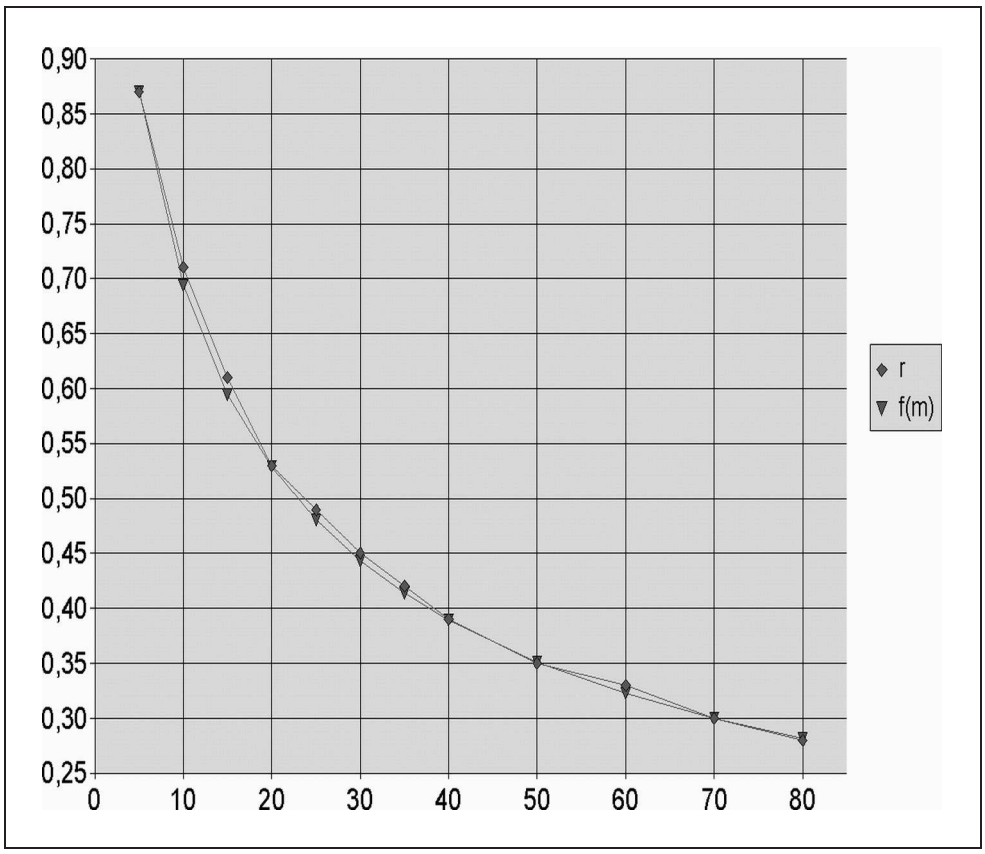

Abb. 4.1: Approximation von $r_{\alpha;m}$ für $\alpha = 0,01$ und $5 \leq m \leq 70$

D.2 Ergebnisse der Regressionsanalyse (S_1 - S_{27})

D.2.1 Ergebnisse der kumulierten Sitzungsdaten

Tab. D.2: Resultate kumulierter Sitzungsdaten (S_1-S_{27})

Rolle R_n	**CLS-Code** c_l	**Stichprobenumfang** N	**Korrelationskoeffizient** $r^a_{n,l}$	**Bestimmtheitsmaß** $B^a_{n,l}$	**Koeffizient** $a^a_{n,l}$	**Koeffizient** $b^a_{n,l}$
1 (IB)	30	70	0,3868	0,1496	-0,9617	0,0564
1 (IB)	29	70	0,3448	0,1189	-2,4185	0,1693
1 (IB)	0	70	0,3427	0,1175	-0,9068	0,0587
1 (IB)	28	70	0,3071	0,0943	-9,9297	0,5861
2 (PL)	39	70	0,3569	0,1274	-0,7733	0,1241
2 (PL)	28	70	0,3375	0,1139	-11,3789	0,702
2 (PL)	2	70	0,331	0,1096	-8,4287	0,5643
3 (BR)	21	70	-0,4224	0,1785	3,171	-0,1251
3 (BR)	24	70	-0,4203	0,1767	7,1574	-0,2736
3 (BR)	7	70	-0,361	0,1303	3,0668	-0,1177
4 (UM)	11	70	0,3914	0,1532	-3,6181	0,2454
4 (UM)	21	70	-0,3659	0,1339	2,5405	-0,1082
4 (UM)	4	70	0,346	0,1197	-4,592	0,3524
4 (UM)	41	70	0,3285	0,1079	-22,9147	1,6347
4 (UM)	20	70	-0,3104	0,0963	1,276	-0,0573
4 (UM)	24	70	-0,3046	0,0928	5,0234	-0,1981
5 (SL)	24	70	-0,337	0,1136	5,0721	-0,1797
6 (PB)	7	70	-0,4227	0,1787	2,4406	-0,106
6 (PB)	24	70	-0,317	0,1005	4,0843	-0,1587
6 (PB)	27	70	0,3064	0,0939	-7,2874	0,6887
7 (FR)	28	70	0,3294	0,1085	-12,9509	0,7244
7 (FR)	11	70	0,313	0,0979	-2,2575	0,1712
7 (FR)	12	70	0,3089	0,0954	-1,034	0,0702
8 (MD)	28	70	0,3324	0,1105	-18,2863	0,8904
8 (MD)	39	70	0,3252	0,1057	-1,7314	0,1456
10 (VP)	4	70	-0,3429	0,1176	10,6367	-0,4392

D.2.2 Ergebnisse der bereinigten kumulierten Sitzungsdaten

Tab. D.3: Resultate bereinigter kumulierter Sitzungsdaten (S_1-S_{27})

Rolle R_n	**CLS-Code** c_l	**Stichprobenumfang** N	**Korrelationskoeffizient** $r^a_{n,l}$	**Bestimmtheitsmaß** $B^a_{n,l}$	**Koeffizient** $a^a_{n,l}$	**Koeffizient** $b^a_{n,l}$
1 (IB)	24	39	-0,4293	0,1843	5,9494	-0,1987
2 (PL)	39	63	0,4999	0,2499	-1,3404	0,165
3 (BR)	32	20	-0,6421	0,4123	8,0343	-0,2926
3 (BR)	24	39	-0,4038	0,163	7,9462	-0,2761
4 (UM)	11	41	0,4226	0,1786	-3,2003	0,2607
4 (UM)	41	56	0,3975	0,158	-28,3518	2,0318
4 (UM)	4	53	0,3497	0,1223	-3,8303	0,347
5 (SL)	47	18	-0,6216	0,3863	7,1942	-0,2262
6 (PB)	30	7	0,918	0,8428	-3,7888	0,2609
6 (PB)	11	41	0,5414	0,2931	-3,0802	0,2777
6 (PB)	24	39	-0,4683	0,2193	6,5459	-0,2451
6 (PB)	27	58	0,364	0,1325	-10,8741	0,9384
7 (FR)	28	19	0,6112	0,3735	-39,4071	2,2279
7 (FR)	29	34	0,4427	0,196	-4,3652	0,3132
7 (FR)	11	41	0,4002	0,1601	-2,691	0,2306
8 (MD)	39	63	0,4516	0,2039	-2,4319	0,1854

D.2.3 Ergebnisse der beteiligungsbezogenen Sitzungsdaten

Tab. D.4: Resultate beteiligungsbezogener Sitzungsdaten (S_1-S_{27})

Rolle R_n	**CLS-Code** c_l	**Stichprobenumfang** N	**Korrelationskoeffizient** $r^c_{n,l}$	**Bestimmtheitsmaß** $B^c_{n,l}$	**Koeffizient** $a^c_{n,l}$	**Koeffizient** $b^c_{n,l}$
1 (IB)	41	70	-0,3065	0,0939	2,5581	-0,0885
3 (BR)	21	70	-0,4183	0,175	0,6001	-0,0244
3 (BR)	10	70	-0,3207	0,1029	0,429	-0,0164
3 (BR)	19	70	-0,3068	0,0941	0,2002	-0,008
5 (SL)	36	70	-0,3176	0,1009	35,1402	-1,0623
5 (SL)	26	70	-0,3096	0,0958	0,2886	-0,0107
5 (SL)	33	70	0,3075	0,0946	-0,0938	0,0053

D.2.4 Ergebnisse der bereinigten beteiligungsbezogenen Sitzungsdaten

Tab. D.5: Resultate bereinigter beteiligungsbezogener Sitzungsdaten (S_1-S_{27})

Rolle R_n	**CLS-Code** c_l	**Stichprobenumfang** N	**Korrelationskoeffizient** $r^c_{n,l}$	**Bestimmtheitsmaß** $B^c_{n,l}$	**Koeffizient** $a^c_{n,l}$	**Koeffizient** $b^c_{n,l}$
4 (UM)	36	47	-0,3833	0,1469	38,8605	-1,0725
4 (UM)	41	56	0,3516	0,1236	-1,5694	0,1334
9 (AR)	29	34	0,5071	0,2571	-1,0237	0,0762

D.2.5 Ergebnisse der zeitbezogenen Sitzungsdaten

Tab. D.6: Resultate zeitbezogener Sitzungsdaten (S_1-S_{27})

Rolle R_n	**CLS-Code** c_l	**Stichprobenumfang** N	**Korrelationskoeffizient** $r^e_{n,l}$	**Bestimmtheitsmaß** $B^e_{n,l}$	**Koeffizient** $a^e_{n,l}$	**Koeffizient** $b^e_{n,l}$
1 (IB)	30	70	0,3885	0,1509	-0,954	0,0558
1 (IB)	0	70	0,3464	0,12	-0,9422	0,0604
1 (IB)	29	70	0,333	0,1109	-2,237	0,1595
1 (IB)	24	70	-0,3077	0,0947	4,7119	-0,1773
1 (IB)	28	70	0,3054	0,0933	-9,7672	0,5765
2 (PL)	28	70	0,334	0,1116	-11,1331	0,6873
2 (PL)	2	70	0,32	0,1024	-7,3461	0,5029
2 (PL)	30	70	0,3076	0,0946	-0,7478	0,0482
3 (BR)	21	70	-0,4164	0,1734	3,2905	-0,1298
3 (BR)	24	70	-0,4081	0,1666	8,0559	-0,311
3 (BR)	7	70	-0,3729	0,1391	3,1706	-0,1227
4 (UM)	11	70	0,3986	0,1589	-3,6725	0,2492
4 (UM)	41	70	0,3675	0,1351	-20,8175	1,492
4 (UM)	21	70	-0,356	0,1268	2,6076	-0,1109
4 (UM)	4	70	0,3314	0,1098	-4,0485	0,3239
4 (UM)	20	70	-0,3085	0,0952	1,2826	-0,0576
5 (SL)	24	70	-0,3142	0,0988	5,5093	-0,1961
6 (PB)	7	70	-0,4141	0,1714	2,4119	-0,1048
6 (PB)	27	70	0,3146	0,099	-7,3444	0,6883
6 (PB)	24	70	-0,3099	0,0961	4,5861	-0,1816
7 (FR)	28	70	0,3272	0,1071	-12,7233	0,7117
7 (FR)	12	70	0,3052	0,0932	-1,0235	0,0696
8 (MD)	28	70	0,3303	0,1091	-17,9681	0,875
10 (VP)	4	70	-0,3681	0,1355	10,8729	-0,4523

D.2.6 Ergebnisse der bereinigten zeitbezogenen Sitzungsdaten

Tab. D.7: Resultate bereinigter zeitbezogener Sitzungsdaten (S_1-S_{27})

n	R_n	c_l	N	$r^e_{n,l}$	$B^e_{n,l}$	$a^e_{n,l}$	$b^e_{n,l}$
1	IB	24	39	-0,4211	0,1774	6,9001	-0,2402
2	PL	39	63	0,4122	0,1699	-0,9582	0,1479
3	BR	32	20	-0,589	0,3469	8,8222	-0,3259
4	UM	41	56	0,4542	0,2063	-25,7672	1,855
4	UM	11	41	0,4355	0,1897	-3,2443	0,2645
5	SL	47	18	-0,6576	0,4324	7,9029	-0,2453
6	PB	30	7	0,908	0,8244	-3,7807	0,2591
6	PB	11	41	0,5172	0,2675	-2,7461	0,2613
6	PB	24	39	-0,4397	0,1934	7,39	-0,2836
6	PB	27	58	0,3749	0,1406	-10,9586	0,9388
7	FR	28	19	0,6065	0,3678	-38,7096	2,1889
7	FR	29	34	0,4449	0,1979	-4,2286	0,3051
8	MD	12	21	0,5669	0,3214	-2,302	0,1557
8	MD	39	63	0,3922	0,1538	-2,1331	0,175

Anhang E

Tabellen zur Problemanalyse

E.1 Legende

In der vorliegenden Arbeit – und speziell in diesem Kapitel – wird immer wieder Bezug genommen auf Benutzertests, die auszugsweise in tabellarischer Form wiedergegeben sind. Dabei findet oftmals eine Filterung (nach Benutzern oder bestimmten CLS++-Codes) statt, um einen ganz bestimmten Sachverhalt zu fokussieren. Die folgende Tabelle E.1 enthält wichtige Hinweise zum Lesen dieser Sitzungsdaten.

Tab. E.1: Legende zu wiedergegebenen VitaminL-Sitzungen

Spalte	**Bedeutung**
Lfd.Nr.	Sämtliche Aktionen aller Benutzer einer Sitzung werden in der Reihenfolge ihres Auftretens auf dem Server erfasst und mit einer laufenden Nummer versehen. Da hierbei die zeitliche Ordnung erhalten bleibt, ist in der Arbeit auch von Zeitpunkten t_i, t_j etc. die Rede
Zeit	Die Spalte *Zeit* gibt den tatsächlichen Zeitpunkt des Eintreffens einer Benutzeraktion auf dem Server wieder. Die Zeitangabe wird dabei relativ zum Sitzungsbeginn im Format `hh:mm:ss` (mit `hh` = Stunden,`mm` = Minuten und `ss` = Sekunden) notiert.
Sender	Die Spalte *Sender* benennt den Auslöser einer Benutzeraktion: Aus datenschutzrechtlichen Gründen sind sämtliche Teilnehmer von Benutzertests in der Form T_i durchnumeriert und somit anonymisiert worden. Zugunsten einer besseren Lesbarkeit werden die *Absender*-Informationen in den entsprechenden Tabellenspalten wiedergegebener Benutzertests üblicherweise nur anhand der Nummer angegeben, d.h. k statt T_k.
Code	Diese Spalte führt den zu einer Benutzeraktion gehörenden CLS++-Code auf (s. Kap.A.1, Tab.A.1).

Fortsetzung auf nächster Seite

Tab. E.1: Legende zu wiedergegebenen VitaminL-Sitzungen *Forts.*

Spalte	**Bedeutung**
Inhalt	In dieser Spalte werden zusätzliche Informationen einer Benutzeraktion veröffentlich. Diese hängen vom Typ der jeweiligen Botschaft ab und beinhalten: • Bei Kommunikationsnachrichten (Code 0 bis 34) wird der Inhalt des verfassten Kommunikationsbeitrags ausgegeben. • Navigationsoperationen mit Code 35 (Dokumentenwechsel) haben die Form $nd \leftarrow ad$, wobei nd das neue, jetzt aktive Dokument und ad das alte, vormals aktive Dokument bezeichnen. • Navigationsoperationen mit Code 36 (Zeilenwechsel) haben die Form $az \rightarrow nz$, wobei az die alte, ehemals aktive Zeile und nz die neue Zeile im aktiven Dokument des auslösenden Benutzers bezeichnen. • Dokumentenoperationen (Code 37ff.) haben die Form $n : N$ mit n = Nummer eines Dokuments innerhalb einer Sitzung und N = Name eines Dokuments (entweder der Dateiname – sofern vorhanden – oder *UNTITLED*x).
Empfänger	Diese Spalte benennt im Fall von Kommunikationsbeiträgen den dedizierten Empfängers des Beitrags, sofern der Absender einen solchen ausgewählt hat. Diese werden analog zu Absenderangaben (s.o.) in der Form T_i notiert.

Nicht alle Tabellen mit Sitzungsdaten machen von allen angegebenen Spalten Gebrauch. Im Einzelfall wird daher zugunsten einer besseren Lesbarkeit auf nicht benötigte Spalten verzichtet.

E.2 Problemmeldungen (S_{69} - S_{77})

E.2.1 Benutzeraktionen zu Problemmeldung 1 (S_{69})

Tab. E.2: Aktionen von Teilnehmer T_{305} im Kontext von Problem 1

Lfd.Nr.	Zeit	Sender	Code	Inhalt
2	0:12:09	305	39	1:UNTITLED1
3	0:12:10	305	36	0 → 1
39	0:16:16	305	28	Zur Ausarbeitung : Hat das evtl mit instance of zu tun? (1. Satz von 2.) .

Fortsetzung auf nächster Seite

Tab. E.2: Aktionen von Teilnehmer T_{305} im Kontext von Problem 1 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
66	0:17:43	305	24	Ich bin nicht sicher, lese gerade mal im skript nach. Wir hatten da neulich was mit Instanz erzeugen. .
101	0:20:45	305	26	Ich denke, wir sollten, d. h. T_{304} sollte dann zweckmäßigerweise velleicht schon mal mit dem anfang anfangen;-), also 1) Getränkeautomat .
416	0:27:06	305	26	Ich denke, wir sollten swing nehmen. das ist doch eigentlich immer netter und hübscher .
461	0:28:31	305	27	Ich denke zwar nein. aber awt ist doch nicht wirklich einfacher oder? .
523	0:29:41	305	28	Zur Ausarbeitung : ja ok, dann machen wir halt so weiter. kein thema. is ja im prinzip wurscht .
528	0:29:43	305	35	3 ← 1
534	0:29:47	305	35	2 ← 3
536	0:29:51	305	35	1 ← 2
537	0:29:52	305	35	3 ← 1
643	0:30:36	305	34	Lasst uns den Tutor fragen !
802	0:31:26	305	12	Kannst Du mir mehr sagen bzw. mal ein beispiel geben zur verwendung von „instance of", bitte *augenaufschlag* ?
867	0:32:12	305	35	2 ← 3
1459	0:40:02	305	29	Lasst es mich so erklären ich suche immer noch danach, wie man „Instanz dieser Klasse erzeugt" umsetzen kann. stelle mich halt wohl blöd an dabei .
1470	0:42:06	305	35	4 ← 2
1471	0:42:19	305	35	2 ← 4
1472	0:42:20	305	35	3 ← 2
1479	0:42:36	305	35	4 ← 3
1481	0:42:42	305	35	3 ← 4
1632	0:45:51	305	35	4 ← 3
1653	0:46:35	305	27	Ich denke nur, dass das ja wohl genauso machen sollen wenn das da steht... .
1753	0:47:47	305	35	5 ← 4
1758	0:48:37	305	35	1 ← 5
1761	0:48:38	305	35	5 ← 1
1762	0:48:39	305	35	4 ← 5
1763	0:48:41	305	35	3 ← 4
1764	0:48:42	305	35	2 ← 3

Fortsetzung auf nächster Seite

Tab. E.2: Aktionen von Teilnehmer T_{305} im Kontext von Problem 1 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1783	0:49:00	305	35	3 ← 2
1848	0:49:26	305	35	2 ← 3
1857	0:49:28	305	35	3 ← 2
2039	0:52:33	305	21	Deshalb höre ich jetzt auch auf zu suchen und werde konstruktiv. habt ihr bestimmte sachen zu deligieren vielleicht? im Übrigen: T_{303}, in setTitle muss ne variable, weil da das gewählte getränk hin soll... .

Tab. E.3: Aktionen aller Teilnehmer im Kontext von Problem 1

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1	0:12:00	304	2	Zusammenfassend wer will was machen? .
2	0:12:09	305	39	1:UNTITLED1
3	0:12:10	305	36	0 → 1
4	0:12:10	304	39	2:UNTITLED2
5	0:12:12	304	35	2 ← 1
6	0:12:12	304	36	0 → 1
7	0:12:15	304	35	1 ← 2
8	0:12:16	304	35	2 ← 1
9	0:13:30	303	26	Ich denke, wir sollten erstma klären was wir alles benötigen .
10	0:14:10	304	27	Ich denke soweit ich das sehe das Automaten-Fenster und das Getränke-Fenster, aber was macht dann der Dritte? .
11	0:15:15	303	27	Ich denke genauso, also muss es wohl eventuell noch mehr geben ;) .
12	0:15:42	304	27	Ich denke ich fang mal mit dem Automaten-Fenster an.Einer von euch kann ja versuchen die Ereignisabhörer einzubauen .
13	0:15:52	304	37	2:UNTITLED2
				⋮
38	0:16:13	304	37	2:UNTITLED2
39	0:16:16	305	28	Zur Ausarbeitung : Hat das evtl mit instance of zu tun? (1. Satz von 2.) .
40	0:16:17	304	36	3 → 1
				⋮
59	0:16:46	304	36	3 → 4

Fortsetzung auf nächster Seite

Tab. E.3: Aktionen aller Teilnehmer im Kontext von Problem 1 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
60	0:16:49	303	6	Verzeihung, aber bitte was? .
61	0:17:28	303	39	3:UNTITLED3
62	0:17:28	303	35	3 ← 1
63	0:17:28	303	36	0 → 1
64	0:17:31	303	35	1 ← 3
65	0:17:32	303	35	2 ← 1
66	0:17:43	305	24	Ich bin nicht sicher, lese gerade mal im skript nach. Wir hatten da neulich was mit Instanz erzeugen. .
67	0:18:38	303	7	Ich sehe, was Du sagen willst, ich les grad auch noch nach .
75	0:18:47	304	36	4 → 2
				⋮
97	0:20:10	303	35	1 ← 2
98	0:20:10	303	35	2 ← 1
99	0:20:11	303	35	3 ← 2
100	0:20:12	303	35	2 ← 3
101	0:20:45	305	26	Ich denke, wir sollten, d. h. T_{304} sollte dann zweckmäßigerweise velleicht schon mal mit dem anfang anfangen;-), also 1) Getränkeautomat .
107	0:21:23	303	4	Ja .
108	0:21:27	303	35	3 ← 2
109	0:22:25	303	35	2 ← 3
110	0:22:26	303	35	3 ← 2
111	0:23:13	303	20	Alternativ dazu wäre es ja nicht so schwer den geträm .
112	0:23:26	303	20	Alternativ dazu getränke frame zu schreiben .
113	0:23:36	303	35	2 ← 3
125	0:23:48	304	36	2 → 3
126	0:23:48	304	36	3 → 4
127	0:23:51	303	35	2 ← 3
128	0:23:52	303	35	3 ← 2
129	0:23:52	304	37	2:UNTITLED2
				⋮
349	0:25:50	304	37	2:UNTITLED2
350	0:26:01	303	6	Verzeihung aber machen wir es nun mit Frame oder JFrame .
				⋮

Fortsetzung auf nächster Seite

Tab. E.3: Aktionen aller Teilnehmer im Kontext von Problem 1 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
416	0:27:06	305	26	Ich denke, wir sollten swing nehmen. das ist doch eigentlich immer netter und hübscher .
417	0:27:36	303	7	Ich sehe, was Du sagen willst aber kommt es immer aufs Äussere an??? .
				⋮
461	0:28:31	305	27	Ich denke zwar nein. aber awt ist doch nicht wirklich einfacher oder? .
462	0:28:57	303	23	Aber T_{304} hat mit awt glaub ich schon angefangen und ich auch nen bissel .
				⋮
523	0:29:41	305	28	Zur Ausarbeitung : ja ok, dann machen wir halt so weiter. kein thema. is ja im prinzip wurscht .
				⋮
538	0:29:58	303	3	Danke schön .
				⋮
643	0:30:36	305	34	Lasst uns den Tutor fragen!
				⋮
802	0:31:26	305	12	Kannst Du mir mehr sagen bzw. mal ein beispiel geben zur verwendung von „instance of", bitte *augenaufschlag*? ?
				⋮
876	0:32:18	304	27	Ich denke ein augenaufschlag wirkt nicht bei maschinen:-) .
				⋮
1283	0:38:38	304	28	Zur Ausarbeitung was macht jetzt eigentlich T_{305}? .
				⋮
1459	0:40:02	305	29	Lasst es mich so erklären ich suche immer noch danach, wie man „Instanz dieser Klasse erzeugt" umsetzen kann. stelle mich halt wohl blöd an dabei .
1460	0:40:41	303	32	Sehr gut !
				⋮
1606	0:45:16	304	24	Ich bin nicht sicher aber was willst du denn mit dieser instance of sache machen? .
				⋮

Fortsetzung auf nächster Seite

Tab. E.3: Aktionen aller Teilnehmer im Kontext von Problem 1 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1653	0:46:35	305	27	Ich denke nur, dass das ja wohl genauso machen sollen wenn das da steht... .
			⋮	
1765	0:48:46	304	27	Ich denke das heisst einfach nur, das dann ein neues fenster aufgeht. dann könnt ihr ja beide getränke fenster machen. sind ja schließlich fünf.
			⋮	
2039	0:52:33	305	21	Deshalb höre ich jetzt auch auf zu suchen und werde konstruktiv. habt ihr bestimmte sachen zu deligieren vielleicht? im Übrigen: T_{303}, in setTitle muss ne variable, weil da das gewählte getränk hin soll... .
2040	0:53:32	303	23	Aber ja, du hast recht das wer ich noch ändern, danke dafür, is mir eben auch aufgefallen .

E.2.2 Benutzeraktionen zu Problemmeldung 2 (S_{69})

Tab. E.4: Kommunikation aller Teilnehmer im Kontext von Problem 2

Lfd.Nr.	Zeit	Sender	Code	Inhalt
2039	0:52:33	305	21	Deshalb höre ich jetzt auch auf zu suchen und werde konstruktiv. habt ihr bestimmte sachen zu deligieren vielleicht? im Übrigen: T_{303}, in setTitle muss ne variable, weil da das gewählte getränk hin soll...
2040	0:53:32	303	23	Aber ja, du hast recht das wer ich noch ändern, danke dafür, is mir eben auch aufgefallen.
2105	0:54:36	304	28	Zur Ausarbeitung dann fang doch mit dem Fenster Bier an.
2118	0:55:40	303	7	Ich sehe, was Du sagen willst also machen wir für jedes getränk nun einen extra frame?? gut wie wollen wir die dann aufteilen??

Fortsetzung auf nächster Seite

Tab. E.4: Kommunikation aller Teilnehmer im Kontext von Problem 2 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
2135	0:56:08	305	24	Ich bin nicht sicher aber die getränkefenster sehen doch alle gleich aus bis auf den titel oder hab ich das falsch verstanden? daher ja der vorschlag mit der variable. im moment mache ich mir gerade gedanken zu der abbruchgeschichte...
2136	0:56:40	304	29	Lasst es mich so erklären achso, willst du dann den abbruch programmieren?können wir auch so machen.
2137	0:56:42	303	11	Weisst Du so hab ich mir das auch gedacht, mehr als der titel ändert sich ja nciht?
2598	1:02:45	304	28	Zur Ausarbeitung T_{305}.willst du mal den Abbruch Button belegen? ich weiß zwar nicht ob das was bringt den in einer extra-klasse zu plazieren, aber dann hast du wenigstens was zu tun.
3039	1:05:23	305	7	Ich sehe, was Du sagen willst ; ich hab gleich die methode; die können wir doch dann bei dir reinkopieren!
3045	1:05:38	304	32	Sehr gut!
3733	1:10:13	304	28	Zur Ausarbeitung wenn du sie fertig hast sag bescheid, damit ich dir dann das dokument freigeben kann.
3744	1:11:06	305	28	Zur Ausarbeitung T_{303}, reicht es nicht, wenn die abbruchmethode an einer stelle steht? is dann nur die frage mit dem zugriff.
3754	1:12:14	303	13	Kannst Du erläutern, warum/wie du das meinst??? ich brauch die abbruch methode ja in meinem frame und so viel zu schreiben is das ja mal nicht, meien können es auch komplizierter machen als es is ;) ?
3785	1:12:56	305	33	Das ist richtig!
3802	1:13:54	303	27	Ich denke dass ich die abbruch methode bei mir shcon drin hab und auf deine nicht zugreifen bruache.
3885	1:15:49	303	1	Lasst mich Euch zeigen fdg.
3924	1:16:44	303	34	Lasst uns den Tutor fragen!

E.2.3 Benutzeraktionen zu Problemmeldung 3 (S_{71})

Tab. E.5: Kommunikation aller Teilnehmer im Kontext von Problem 3

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
4	0:01:45	291	2	Zusammenfassend wie gesagt... .	0
5	0:01:49	293	0	Okay, lasst uns fortfahren .	0
9	0:02:11	292	2	Zusammenfassend ich glaub, ich hab da schon was gefunden .	0
12	0:02:27	293	15	Denkst Du awt oder swing ?	0
13	0:03:11	291	20	Alternativ dazu nehmen wir doch einfach swing .	0
14	0:03:15	292	27	Ich denke awt, für aufgabe 1 mit dem gridlayout .	0
15	0:03:43	293	27	Ich denke auch auf jeden fall grid layout .	0
16	0:03:47	291	28	Zur Ausarbeitung werde ich dann mal da inet benutzen..grrr hoffentlich geht das .	0
17	0:04:10	291	31	Ich bin ziemlich sicher zunächst brauchen wir nur jFrame .	0
18	0:04:26	293	11	Weisst Du ob wir in beiden klassen das gleich benutzen müssen ?	0
19	0:04:27	292	2	Zusammenfassend ich geh auch mal ins netz und kopiere das programm .	0
21	0:05:01	291	28	Zur Ausarbeitung was machst du jetzt T_{293}? .	0
23	0:05:32	293	27	Ich denke ich mach erstmal etwas bei T_{292} mit, ok? .	0
24	0:05:51	291	13	Kannst Du erläutern, warum/wie man das internet startet bei gehts nicht ?	0
32	0:06:46	291	30	Zur Rechtfertigung habs hingekriegt .	0
33	0:07:10	293	11	Weisst Du ob ich das dokument jetzt schon hab? könnte ja schon mal anfangen solamge du suchst ?	292
34	0:07:47	292	28	Zur Ausarbeitung ich habs schon gefunden, muss es nur noch einfügen .	293
38	0:08:22	293	30	Zur Rechtfertigung ic hab doch jetzt aber nichts zu tun .	292

Fortsetzung auf nächster Seite

Tab. E.5: Kommunikation aller Teilnehmer im Kontext von Problem 3 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
42	0:09:12	292	6	Verzeihung aber wie füge ich jetzt das programm ein??? .	0
53	0:10:25	291	24	Ich bin nicht sicher aber irgendwie kann man hier nichts einfügen .	0
72	0:10:43	292	6	Verzeihung wie füge ich das programm ein, es geht nicht .	293
94	0:11:09	293	31	Ich bin ziemlich sicher das du da lieber T_{284} fragen solltest .	292
153	0:11:38	292	6	Verzeihung wie kann ich hier programme aus dem internet einfügen? .	284
166	0:13:28	284	2	Zusammenfassend : Daten am besten auf Desktopn speichern, wenn nötig entzippen und dann ins Programm importieren .	0
171	0:14:59	293	12	Kannst Du mir mehr sagen was du jaetzt überhaupt machst., vierlleicht sollten wir erstmal einen groben plan über das programm erstellen, weil so fühle ich mich gerade sehr unnütz ?	0
178	0:16:25	292	2	Zusammenfassend ich habe jetzt grad ein programm eingefügt, mit gridlayout, das müssen wir doch jetzt eigentlich nur noch umbenennen und ändern, oder? .	293
181	0:17:02	293	30	Zur Rechtfertigung bei mir ist aber bei dem programm immer noch nichts zu sehen .	292
183	0:17:31	293	30	Zur Rechtfertigung ach so, du hast es ja ganz neu erstellt .	0
195	0:18:28	292	2	Zusammenfassend ja, jetzt müssen wir es noch umbenennen und das andere löschen, weißt du, wie? ich gebs dir mal frei .	293
205	0:18:56	291	23	Aber bei mir geht das mit dem einfügen auch nicht wirklich ich glaube ich lase es deshalb auch .	0
244	0:19:49	291	27	Ich denke ihr habt da grade bei mir was eingefügt oder .	0

Fortsetzung auf nächster Seite

Tab. E.5: Kommunikation aller Teilnehmer im Kontext von Problem 3 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
258	0:20:10	291	30	Zur Rechtfertigung hab falsch geguckt sorry .	0
291	0:20:41	291	31	Ich bin ziemlich sicher ihr könnt mir jetzt erklären wie ihr das gemacht habt .	0
383	0:21:45	292	29	Lasst es mich so erklären wir haben das programm aus dem internet geladen, eingefügt und umbenannt und jetzt können wir es verändern .	0
388	0:22:23	291	29	Lasst es mich so erklären ich kriege das gar nicht hin .	0
394	0:23:00	293	24	Ich bin nicht sicher aber wollen wir erstmal compilieren .	0
396	0:23:16	292	4	Ja .	293
397	0:23:31	293	11	Weisst Du wie das nochmal hier ging ?	0
454	0:25:33	292	2	Zusammenfassend also, du musst erst den ganzen ordner von der java seite laden, also das ganze „Aufgaben und Beispiele" oder wie das heißt. den speicherst du und dann kannst du dir das programm, das sdu brauchst und extrahierst es und dann gehst du in vitaminl auf datei importieren und kannst es dann raussuchen und einfügen .	291
461	0:25:48	284	6	Verzeihung das compiliern-Symbol ist das unten links in der Werkzeuganzeige, das daneben ist zum Ausführen .	293
464	0:26:14	293	3	Danke schön .	284
465	0:26:17	291	3	Danke schön .	0
503	0:27:20	292	28	Zur Ausarbeitung gehts mit dem compilieren? .	293
516	0:27:43	293	13	Kannst Du erläutern, warum/wie jetzt fehler angezeigt werden ?	292
518	0:28:23	292	11	Weisst Du ich sehe gar nichts ?	293
519	0:28:51	293	27	Ich denke ich gebe es dir mal frei .	292

Fortsetzung auf nächster Seite

Tab. E.5: Kommunikation aller Teilnehmer im Kontext von Problem 3 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
525	0:29:16	291	22	Wenn ich in meinem programm als titel das in eurem programm gewählte getraenk habe Dann müsst ihr mir mal eure methodennamen sagen für die getraenkewahl!!!!!!!!!!!!!!!!!!!!.	0
558	0:29:57	291	28	Zur Ausarbeitung hab ich das ganze einfach mal getraenk genannt .	0
562	0:30:03	293	19	Ich stimme nicht zu, weil wir noch gar nicht so weit sind .	0
614	0:30:47	292	12	Kannst Du mir mehr sagen wie ich das hinkriege, das er mir fehler anzeigt? ?	293
671	0:31:30	293	13	Kannst Du erläutern, warum/wie du du das getränk als titel brauchst? deine klasse soll doch bei jedem der getränke aufgerufen werden, oder ?	291
690	0:31:53	293	29	Lasst es mich so erklären erst compilieren, dann ausführen .	292
698	0:33:00	292	28	Zur Ausarbeitung kommt bei dir dann auch erst so ein kasten, wo was eingeben musst .	293
699	0:33:09	291	7	Ich sehe, was Du sagen willst aber nichtsdesdotrotz (?) muss mein titel mit der getraenkewahl übereinstimmen .	0
700	0:33:11	284	27	Ich denke ihr solltet vor dem Compilieren an die Endung .java denken ;-) .	0
722	0:33:25	293	29	Lasst es mich so erklären da braust du nur AutomatFrame eingeben .	292
726	0:33:35	293	3	Danke schön .	284
772	0:34:49	292	6	Verzeihung ich gebe dir das programm mal frei, hast du die endung java dahintergepackt? .	293
789	0:35:08	293	30	Zur Rechtfertigung können den namen ja später immer noch anpassen .	291
884	0:37:00	292	6	Verzeihung klappt es jetzt mit dem compiler? .	293

Fortsetzung auf nächster Seite

Tab. E.5: Kommunikation aller Teilnehmer im Kontext von Problem 3 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
885	0:37:03	293	9	Würdest Du bitte auch mal compilieren? hatte .java vergessen, aber es werden immer noch fehler angezeigt ?	292
889	0:38:03	293	27	Ich denke du brauchst es dafür nicht, aber ich kann es natürlich trotzdem gerne freigeben ;) .	292
894	0:38:32	291	6	Verzeihung aber ich weiß gar nicht wie ich das programm ausführe .	284
935	0:39:13	293	15	Denkst Du wir sollten mal den tutor fragen wegen dem error ?	292
939	0:39:24	292	4	Ja .	293
940	0:39:31	293	34	Lasst uns den Tutor fragen !	0
941	0:39:57	284	21	Deshalb musst du nach dem erfolgreichen Compilieren auf das Symbol unten rechts in der Werkzeuganzeige klicken, um das Programm auszuführen, probier's mal! .	291
947	0:40:56	293	11	Weisst Du woran es liegt, dass der Compiler bei AutomatFrame.java einen Fehler anzeigt ?	306

Tab. E.6: Aktionen von Teilnehmer T_{293} im Kontext von Problem 3

Lfd.Nr.	Zeit	Sender	Code	Inhalt
5	0:01:49	293	0	Okay, lasst uns fortfahren .
12	0:02:27	293	15	Denkst Du awt oder swing?
15	0:03:43	293	27	Ich denke auch auf jeden fall grid layout.
18	0:04:26	293	11	Weisst Du ob wir in beiden klassen das gleich benutzen müssen ?
20	0:04:51	293	44	1:Getränkeautomat.java
23	0:05:32	293	27	Ich denke ich mach erstmal etwas bei T_{292} mit, ok? .
33	0:07:10	293	11	Weisst Du ob ich das dokument jetzt schon hab? könnte ja schon mal anfangen solamge du suchst ?
38	0:08:22	293	30	Zur Rechtfertigung ic hab doch jetzt aber nichts zu tun .
49	0:09:33	293	35	$2 \leftarrow 1$
50	0:09:34	293	35	$1 \leftarrow 2$

Fortsetzung auf nächster Seite

Tab. E.6: Aktionen von Teilnehmer T_{293} im Kontext von Problem 3 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
94	0:11:09	293	31	Ich bin ziemlich sicher das du da lieber T_{306} fragen solltest .
171	0:14:59	293	12	Kannst Du mir mehr sagen was du jaetzt überhaupt machst., vierlleicht sollten wir erstmal einen groben plan über das programm erstellen, weil so fühle ich mich gerade sehr unnütz ?
181	0:17:02	293	30	Zur Rechtfertigung bei mir ist aber bei dem programm immer noch nichts zu sehen .
182	0:17:07	293	35	$3 \leftarrow 1$
183	0:17:31	293	30	Zur Rechtfertigung ach so, du hast es ja ganz neu erstellt .
192	0:18:21	293	35	$1 \leftarrow 3$
193	0:18:22	293	35	$2 \leftarrow 1$
194	0:18:24	293	35	$1 \leftarrow 2$
198	0:18:40	293	35	$3 \leftarrow 1$
203	0:18:48	293	35	$3 \leftarrow 0$
204	0:18:54	293	44	3:GridLayoutJFrame.java
206	0:19:40	293	42	3:GetraenkFrame
207	0:19:44	293	36	$1 \rightarrow 6$
				$\vdots$
394	0:23:00	293	24	Ich bin nicht sicher aber wollen wir erstmal compilieren .
395	0:23:06	293	36	$36 \rightarrow 39$
397	0:23:31	293	11	Weisst Du wie das nochmal hier ging ?
403	0:23:42	293	36	$39 \rightarrow 51$
417	0:24:06	293	36	$51 \rightarrow 36$
443	0:24:57	293	35	$3 \leftarrow 4$
460	0:25:44	293	36	$36 \rightarrow 13$
462	0:25:55	293	47	3:AutomatFrame
464	0:26:14	293	3	Danke schön .
466	0:26:21	293	47	3:AutomatFrame
483	0:26:42	293	48	0:null
516	0:27:43	293	13	Kannst Du erläutern, warum/wie jetzt fehler angezeigt werden ?
517	0:28:23	293	36	$13 \rightarrow 41$
519	0:28:51	293	27	Ich denke ich gebe es dir mal frei .
520	0:28:57	293	41	3:AutomatFrame
521	0:28:58	293	45	3:AutomatFrame

Fortsetzung auf nächster Seite

Tab. E.6: Aktionen von Teilnehmer T_{293} im Kontext von Problem 3 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
562	0:30:03	293	19	Ich stimme nicht zu, weil wir noch gar nicht so weit sind .
671	0:31:30	293	13	Kannst Du erläutern, warum/wie du du das getränk als titel brauchst? deine klasse soll doch bei jedem der getränke aufgerufen werden, oder ?
690	0:31:53	293	29	Lasst es mich so erklären erst compilieren, dann ausführen .
694	0:32:02	293	47	3:AutomatFrame
695	0:32:07	293	48	0:null
722	0:33:25	293	29	Lasst es mich so erklären da braust du nur AutomatFrame eingeben .
726	0:33:35	293	3	Danke schön .
789	0:35:08	293	30	Zur Rechtfertigung können den namen ja später immer noch anpassen .
801	0:35:18	293	44	3:AutomatFrame
818	0:35:33	293	42	3:AutomatFram.java
836	0:35:47	293	42	3:AutomatFrame.java
842	0:35:52	293	47	3:AutomatFrame.java
885	0:37:03	293	9	Würdest Du bitte auch mal compilieren? hatte .java vergessen, aber es werden immer noch fehler angezeigt?
889	0:38:03	293	27	Ich denke du brauchst es dafür nicht, aber ich kann es natürlich trotzdem gerne freigeben ;) .
890	0:38:05	293	45	3:AutomatFrame.java
935	0:39:13	293	15	Denkst Du wir sollten mal den tutor fragen wegen dem error ?
940	0:39:31	293	34	Lasst uns den Tutor fragen !
947	0:40:56	293	11	Weisst Du woran es liegt, dass der Compiler bei AutomatFrame.java einen Fehler anzeigt?

E.2.4 Benutzeraktionen zu Problemmeldung 4 (S_{74})

Tab. E.7: Kommunikation aller Teilnehmer im Kontext von Problem 4

Lfd.Nr.	Zeit	Sender	Code	Inhalt
109	0:04:42	286	11	Weisst Du womit wir anfangen sollen ?

Fortsetzung auf nächster Seite

Tab. E.7: Kommunikation aller Teilnehmer im Kontext von Problem 4 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
168	0:05:13	296	24	Ich bin nicht sicher .
204	0:05:37	286	27	Ich denke T_{294} schafft das alles alleine .
205	0:05:58	296	12	Kannst Du mir mehr sagen sag mal was wir machen sollen?und teil das ein ?
206	0:07:04	286	27	Ich denke es gibt in Java nen Sortier Algorythmus und man muss ihm nur mit dem Interface sagen wie er was zu sortieren hat .
207	0:07:08	294	2	Zusammenfassend wie ging das mit bubble sort nochmal? .
209	0:07:48	286	27	Ich denke du suchst das ganze Array nach dem kleinsten Ab schreibst es an pos 1, gehst zu pos 2 und suchst dann das 2te verbleibenden .
217	0:08:22	294	3	Danke schön .
276	0:09:26	286	27	Ich denke T_{296} udn ich sollten vllt mal mit Read and Write anfangen .
402	0:11:28	286	27	Ich denke T_{296} hat deinen Bubblesort gleich fertig :-P .
432	0:11:58	296	6	Verzeihung T_{294} auch .
441	0:12:00	286	27	Ich denke T_{296} udn ich machen mit Readand-Write weiter ;-) .
457	0:12:11	296	4	Ja .
518	0:12:27	296	6	Verzeihung und wie , was soll ich tun? .
582	0:13:50	286	27	Ich denke wir brauchen eine statische METhode die alles einliest, aus der Datei heraus und eine die alles aus dem Array in die Textdatei einliest.
583	0:13:57	294	2	Zusammenfassend gesprochen, uns gehts gut, und dir? .
635	0:14:26	296	7	Ich sehe, was Du sagen willst kannst du das denn? .
841	0:15:31	286	27	Ich denke nicht, aber is ja wohl kein Prob es zu lernen ;-) .
968	0:17:26	286	9	Würdest Du bitte mir nochmal den Dateinamen zum schreiben sagen, T_{296} ?
971	0:17:42	286	27	Ich denke ich habs gefunden .
1003	0:18:34	286	34	Lasst uns den Tutor fragen !
1004	0:18:41	296	6	Verzeihung public FileReader(String fileName).
1005	0:18:59	286	27	Ich denke danke .

Fortsetzung auf nächster Seite

Tab. E.7: Kommunikation aller Teilnehmer im Kontext von Problem 4 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1038	0:19:25	294	27	Ich denke du solltes noch io.filereader und iofilewriter importieren .
1044	0:19:27	296	6	Verzeihung ,,,public FileWriter(String fileName) .
1101	0:20:45	286	27	Ich denke ich schau mal kurz in die Api ;-) .
1103	0:21:05	296	6	Verzeihung was suchst du denn? .
1104	0:21:18	296	6	Verzeihung was suchst du denn .
1108	0:21:37	286	27	Ich denke wie ich Zeile für Zeile auslese .
1112	0:22:01	294	27	Ich denke du solltest nen leseString schreiben, der den dateinamen einließt, zumindest versteh ich die aufgabe so .
1113	0:22:18	296	6	Verzeihung liest der nicht die gesamte datei ein? .
1122	0:22:48	286	27	Ich denke dass, das in der in der Mainmehtode einfach angegeben wird. .
1156	0:23:15	286	27	Ich denke ja, nur soll das ja Zeile für Zeile in neue Strings geschrieben werden, in ein sTringArray .
1367	0:26:22	296	6	Verzeihung woher weis er eigentlich wo er die datei suchen mus,ist die im selben verzeichnis? .
1368	0:26:44	286	27	Ich denke ja, da ich sie da hinnein kopiert hab ;-) .
1369	0:27:08	286	27	Ich denke das mit dem lesen Zeile für Zeile kopier ich mal so aus der Vorlesung, da es da genauso vorhanden is wie wir das brauchen ;-) .

Tab. E.8: Kommunikation aller Teilnehmer nach Problem 4

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1577	0:26:17	294	11	Weisst Du ob man noch irgendwas ändern musste ausser dem Datentypen, um das Array zu sortieren? ?
1960	0:32:40	286	27	Ich denke ich hab keine Ahnung .
2044	0:34:04	294	2	Zusammenfassend ich auch nicht, in er api steht leider auch nix, zumindest hab ich noch nix gefunden .
2081	0:34:41	286	23	Aber haste schon mal unter Compareable gesucht? Müsste n Interface sein, also krusiv geschrieben .

Fortsetzung auf nächster Seite

Tab. E.8: Kommunikation aller Teilnehmer nach Problem 4 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
2446	0:40:46	286	27	Ich denke danke :o) .
2588	0:43:08	286	27	Ich denke ich habe fertig .
2610	0:43:30	296	6	Verzeihung soll die sortierte liste auch abgespeichert werden? .
2647	0:44:07	286	27	Ich denke du musst noch n Compare schreiben .
2675	0:44:35	294	25	Mit anderen Worten is erledigt .
2789	0:46:29	286	13	Kannst Du erläutern, warum/wie das mit dem vergleichen zweier Strings so funktioniert??? ?
2790	0:46:30	294	2	Zusammenfassend ich denke ja, warum?? .
2987	0:49:47	286	27	Ich denke du musst noch die MEthode Swap schreiben .
3682	1:01:22	294	6	Verzeihung , und nun testlauf?? .
3685	1:01:25	286	27	Ich denke ich korrigiere erstmal meine 10 Fehler ;-) .
4459	1:14:19	296	6	Verzeihung lassen strings sich so vergleichen? .
4460	1:14:20	286	27	Ich denke ich bin dumm, kann mir mal bitte wer erklären wieso ich das erste mal auf datei zugreifen kann und er beim zweiten mal meint, as Datei evtl. nicht initialisiert wurde? .
4461	1:14:21	294	2	Zusammenfassend anscheinend ja nicht, sonst wärs ja kein fehler .
4792	1:19:52	296	6	Verzeihung und hast du denn fehler gefunden? .
4904	1:21:44	286	27	Ich denke ich würd sagen es funzt soweit alles .
4916	1:21:56	286	27	Ich denke oder auch nicht, die Sortnamen is ja noch leer :-(.
5534	1:32:14	286	27	Ich denke wie wärs wenn du nur das erste Char im Wort miteiander vergleichst? .
5537	1:32:17	286	27	Ich denke String.charAt(0) .
5565	1:32:45	286	26	Ich denke, wir sollten du kannst n char in nen int umwandeln .
5645	1:34:05	294	2	Zusammenfassend versteh ich nicht so ganz, habs mal freigegeben .
5739	1:35:39	286	27	Ich denke so könnts gehen .
5799	1:36:39	294	2	Zusammenfassend und der weiß jetzt selbst, das a=1, b=2,...oder wie läuft das? .

Fortsetzung auf nächster Seite

Tab. E.8: Kommunikation aller Teilnehmer nach Problem 4 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
5801	1:36:41	286	2	Zusammenfassend ne a = 65 b = 66 usw halt Uniciode/ASCII Tabelle .
5807	1:36:47	294	0	Okay, lasst uns fortfahren .
6226	1:43:46	286	27	Ich denke wenn wir jetzt herraus finden wieso der net abspeichert sind wir fertig .

E.2.5 Benutzeraktionen zu Problemmeldung 5 (S_{76})

Tab. E.9: Kommunikation aller Teilnehmer im Kontext von Problem 5

Lfd.Nr.	Zeit	Sender	Code	Inhalt
4	0:05:50	304	2	Zusammenfassend also wer will was machen?.
5	0:06:13	304	2	Zusammenfassend also wer will was machen?.
6	0:06:22	305	15	Denkst Du ich soll readandwrite machen?
7	0:07:02	303	24	Ich bin nicht sicher aber ich les mir das grad nochmal alles durch, da ich damit noch nicht so ganz vuiel gemacht hab .
8	0:08:03	305	25	Mit anderen Worten es wär vielleicht gut wenn T_{303} ablauf macht... .
9	0:08:35	304	11	Weisst Du ok, dann macht T_{305} readandwrite, T_{303} ausgabe und ich sortieren, okay? ?
13	0:09:02	305	4	Ja.
14	0:09:20	303	4	Ja.
15	0:09:27	305	31	Ich bin ziemlich sicher dass vor dem punkt kein leerzeichen stehen darf .
904	0:25:49	305	28	Zur Ausarbeitung T_{304}, in den beispielen zu v12 is doch bestimmt was, was du reinkopieren kannst... hab mich da auch schon kräftig bedient.
920	0:27:04	304	6	Verzeihung ich bin da auch schon am gucken bei Bibliotheksortieren, aber ich muss ja erst mal rausfinden was ich da ändern muss damit es bei der aufgabe passt.
927	0:28:18	304	10	Entschuldigt mich wie heißt denn z.B. das Array was ich sortieren soll? leseDatei[ ] oder wie? .

Fortsetzung auf nächster Seite

Tab. E.9: Kommunikation aller Teilnehmer im Kontext von Problem 5 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
939	0:30:20	304	27	Ich denke ich finde bisher nur sortierfunktionen für int. Es sei denn die Daten die in deiner Textdatei sind sind comparable, dann könnte das vielleicht auch klappen .
959	0:33:12	305	24	Ich bin nicht sicher , aber eigentlich müssten strings schon vergleichbar sein. und mit dem array: ja, da muss ich mich noch drum kümmern, die strings in ein solches zu stecken. jedenfalls soll laut aufgabenstellung meine methode ein array ausspucken .
1018	0:35:11	304	11	Weisst Du ich bin grad ziemlich ratlos. in Bibliotheksortieren wird einmal sortieren.sort aufgerufen, aber ich finde da keine methode sort in sortieren. wär vielleicht schon gut wenn ich wüsste wo dieser befehl herkommt?
1021	0:36:38	305	9	Würdest Du bitte sagen wo genau du meinst ?
1022	0:37:50	304	27	Ich denke bibliotheksortieren //und nun das sortieren aller medien// dahinter .
1027	0:38:55	303	2	Zusammenfassend die methode sort is in der klasse sortieren .
1028	0:39:18	304	11	Weisst Du wo da? ich finde es da nicht. ?
1032	0:39:38	305	10	Entschuldigt mich , aber ich muss mal für kleine programmierer .
1082	0:41:04	303	7	Ich sehe, was Du sagen willst wenn du die klasse sortieren aufrufst drück strg-f und such nach „sort”, wordpad hat leider keine zeilen angaben.
1191	0:43:07	303	15	Denkst Du du hast sie gefunden???
1195	0:43:19	304	30	Zur Rechtfertigung sort gibt es in sortieren nicht. ich versuche einfach bubble sort und entweder es funktioniert mit strings oder nicht.
1253	0:44:10	303	6	Verzeihung schauen wir auch in der selben klasse sortieren nach?? is die letzte oder vorletzte methode in sortieren.
1577	0:46:34	303	2	Zusammenfassend in v12.zip gibt es 2 mal sortieren, eines davon gehört zur bibliothek und enthält auch das sort.
1624	0:47:20	304	3	Danke schön.

Fortsetzung auf nächster Seite

Tab. E.9: Kommunikation aller Teilnehmer im Kontext von Problem 5 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
2063	0:52:46	304	27	Ich denke ich übergebe euch einfach sortieren und dann muss in T_{303}s klasse das array zum sortieren festgelegt werden. bei mir heißt es m, also muss es noch mit dem array gleichgesetzt werden wo die daten drinsind .
2180	0:54:33	304	2	Zusammenfassend T_{303} muss wohl erst lesedatei aufrufen, dann den kram in ein array, dann sortieren und dann dateischreiben, oder? .
2226	0:57:29	304	11	Weisst Du T_{303}, willst du mal kreativ sein und eine datei namen.txt erstellen und dir ganz viele tolle namen ausdenken ?
2228	0:57:45	305	24	Ich bin nicht sicher ob das so richtig ist, aber: ich mach nu ne methode, die die namen liest und ein stringarray namens arrayunsorted ausgibt. meine zweite methode schreibenDatei braucht das stringarray arraysorted und schreibt in die textdatei.
2230	0:58:15	303	25	Mit anderen Worten könnt ihr mir mal sagen was ich nun machen muss??? .
2244	0:59:08	304	6	Verzeihung so hab ich das auch verstanden, es muss also dein arrayunsorted =m gesetzt werden und dann die sortiermethode aufgerufen werden. .
2345	1:00:04	304	21	Deshalb T_{303} musst du erst die methode dateilesen aufrufen, dann m = arrayunsorted, dann sortieren und dann dateischreibe. die namen mach ich dann .
2507	1:01:47	304	25	Mit anderen Worten als erstes ReadAndWrite.leseDatei(); .
2713	1:03:22	304	27	Ich denke T_{303} sollte die befehle langsam mal einbauen .
2783	1:03:56	303	7	Ich sehe, was Du sagen willst aber T_{303}, hängt grad übelst, und bekommt selbst das nicht hin, ich hab kein plan was los is .
2788	1:04:27	304	11	Weisst Du dann gib das dokument mal frei, dann zeige ich dir was ich meine ?
2789	1:04:36	303	7	Ich sehe, was Du sagen willst danke dafür .
2795	1:05:12	303	21	Deshalb is es jezzt frei .
2828	1:06:14	304	11	Weisst Du jetzt muss ich erst mal gucken wie ich das dann übernehme ?

Fortsetzung auf nächster Seite

Tab. E.9: Kommunikation aller Teilnehmer im Kontext von Problem 5 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
2925	1:07:16	303	27	Ich denke „gut jetzt hast du es".
2972	1:08:31	304	27	Ich denke so müsste das funktionieren. jetzt müssen noch die anderen befehle aufgerufen werden .
2996	1:08:41	303	24	Ich bin nicht sicher aber musst nciht noch angeben das arrayunsorted in readandwrite is? .
3037	1:09:43	304	27	Ich denke wir können ja mal beides versuchen. bastel erst mal alle befehle rein und dann compilier mal. eins müsste ja wohl richtig sein. .
3081	1:11:04	304	28	Zur Ausarbeitung noch ein; und dann die nächste methode .
3090	1:11:50	305	6	Verzeihung kann es sein, dass ich zwei mal lesen muss, weil ich das string array ja erst erzeugen muss, bevor ich was reinschreib, aber zur erzeugung brauche ich ja die anzahl der elemente, also die zeilenzahl...
3091	1:11:57	305	34	Lasst uns den Tutor fragen !
3092	1:12:07	303	10	Entschuldigt mich aber verbesseer bitte alles wenn es was gibt .
3100	1:12:22	304	33	Das ist richtig !
3107	1:13:13	305	16	Bitte zeige mir ob es möglich ist, die zeilenanzahl in einer datei einfach so zu bestimmen!
3143	1:13:53	304	27	Ich denke du musst erst noch m = arraysorted setzen .
3145	1:14:23	303	6	Verzeihung wer jetzt? ich??? .
3148	1:14:30	304	33	Das ist richtig !
3152	1:14:48	303	3	Danke schön .
3191	1:15:25	305	28	Zur Ausarbeitung ok, also ab jetzt die namenzahl nicht mehr ändern. das array hat halt ne länge von 10 und feddich.

Tab. E.10: Aktionen von Teilnehmer T_{305} im Kontext von Problem 5

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1255	0:44:11	305	37	1:ReadAndWrite
1257	0:44:12	305	37	1:ReadAndWrite
1258	0:44:12	305	37	1:ReadAndWrite

Fortsetzung auf nächster Seite

Tab. E.10: Aktionen von Teilnehmer T_{305} im Kontext von Problem 5 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
		⋮		
3035	1:09:35	305	37	1:ReadAndWrite
3036	1:09:36	305	37	1:ReadAndWrite
3038	1:09:46	305	36	75 → 74
3039	1:09:49	305	36	74 → 75
3058	1:10:18	305	41	1:ReadAndWrite
3090	1:11:50	305	6	Verzeihung kann es sein, dass ich zwei mal lesen muss, weil ich das string array ja erst erzeugen muss, bevor ich was reinschreib, aber zur erzeugung brauche ich ja die anzahl der elemente, also die zeilenzahl...
3091	1:11:57	305	34	Lasst uns den Tutor fragen !
3107	1:13:13	305	16	Bitte zeige mir ob es möglich ist, die zeilenanzahl in einer datei einfach so zu bestimmen!

E.2.6 Benutzeraktionen zu Problemmeldung 6 (S_{76})

Tab. E.11: Kommunikation aller Teilnehmer im Kontext von Problem 6

Lfd.Nr.	Zeit	Sender	Code	Inhalt
3538	1:23:08	305	28	Zur Ausarbeitung ok, lasst uns mal testen. habs freigegeben .
3539	1:23:10	304	10	Entschuldigt mich ich habs mal compiliert und T_{303}s vorschlag war richtig, dass man sagen muss woher die arraynamen kommen .
3547	1:23:56	303	21	Deshalb ich geb meins mal frei dann kannst das beseitigen wenn du weißt wie .
3549	1:24:31	304	6	Verzeihung jemand ne idee?
3551	1:24:52	305	2	Zusammenfassend e fehler vomcompiler sehe ich nicht. wat mach ich schussel denn nu wieder falsch?
3555	1:25:26	304	2	Zusammenfassend vielleicht leseDatei.arrayunsorted?
3556	1:25:54	304	28	Zur Ausarbeitung vielleicht siehst du die fehler nur wenn du selber compilierst?
3586	1:26:49	304	24	Ich bin nicht sicher aber so vielleicht?
3589	1:27:06	303	7	Ich sehe, was Du sagen willst.

Fortsetzung auf nächster Seite

Tab. E.11: Kommunikation aller Teilnehmer im Kontext von Problem 6 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
3590	1:27:17	305	24	Ich bin nicht sicher aber muss man namen von arrays nicht immer mit [ ] dahinter angeben...? T_{304}, naja, ich hab bei mir auf das compiliersymbol geklickt und im compilerfenster steht nix. im systemfenster steht, dass ich compiliert habe und es fehler gab, aber nich welche.
3591	1:27:17	304	25	Mit anderen Worten jetzt hab ich schon 8 fehler.
3609	1:27:59	304	15	Denkst Du so?
3641	1:29:16	303	25	Mit anderen Worten jetzt sind es nur noch 6 fehler.
3643	1:29:55	305	34	Lasst uns den Tutor fragen!
3644	1:30:04	304	4	Ja.
3645	1:30:27	305	13	Kannst Du erläutern, warum/wie ich irgendwie nich rauskriege, wo die fehler in meinem quelltext sind?
3655	1:31:31	304	6	Verzeihung nur noch 5 fehler.
3656	1:31:32	303	7	Ich sehe, was Du sagen willst T_{305} ich hatte das problem letzte woche auch als ich vergessen hatte beim speichern .java hinter die datei zu setzen...
3657	1:31:44	305	13	Kannst Du erläutern, warum/wie ich die fehler beim compilieren sehen kann?
3665	1:32:12	304	6	Verzeihung nur noch 4.
3687	1:32:37	305	22	Wenn da unten bei dateityp java steht Dann denke ich doch, dass das nich not tut. aber dann mal danke T_{303}! werde ich sofort ändern.
3691	1:33:29	303	21	Deshalb , ich dahcte das letzte mal auch, aber musste ich noch angeben.
3696	1:33:45	304	6	Verzeihung vielleicht will es mal jemand von euch versuchen?
3726	1:35:12	304	6	Verzeihung nun nur noch 2.
3783	1:36:48	305	9	Würdest Du bitte sagen welche frage das war? das mit dem compiler wird ja wohl an der dateiendung liegen, also das hat sich erledigt. dafür hab ich jetzt das problem, dass ich nicht weiß, wie ich meine letzte version herholen und neu abspeichern soll ?

Tab. E.12: Aktionen von Teilnehmer T_{305} im Kontext von Problem 6

Lfd.Nr.	Zeit	Sender	Code	Inhalt
3192	1:15:31	305	36	75 → 72
3196	1:15:37	305	36	72 → 40
			⋮	
3535	1:22:40	305	41	1:ReadAndWrite
3536	1:22:45	305	45	1:ReadAndWrite
3538	1:23:08	305	28	Zur Ausarbeitung ok, lasst uns mal testen. habs freigegeben.
3542	1:23:20	305	47	1:ReadAndWrite
3551	1:24:52	305	2	Zusammenfassend e fehler vom compiler sehe ich nicht. wat mach ich schussel denn nu wieder falsch?
3552	1:25:09	305	35	2 ← 1
3553	1:25:10	305	35	3 ← 2
3590	1:27:17	305	24	Ich bin nicht sicher aber muss man namen von arrays nicht immer mit [ ] dahinter angeben...? T_{304}, naja, ich hab bei mir auf das compiliersymbol geklickt und im compilerfenster steht nix. im systemfenster steht, dass ich compiliert habe und es fehler gab, aber nich welche .
3612	1:28:17	305	35	1 ← 3
3613	1:28:18	305	47	1:ReadAndWrite
3614	1:28:35	305	35	2 ← 1
3615	1:28:38	305	35	1 ← 2
3643	1:29:55	305	34	Lasst uns den Tutor fragen !
3645	1:30:27	305	13	Kannst Du erläutern, warum/wie ich irgendwie nich rauskriege, wo die fehler in meinem quelltext sind?
3657	1:31:44	305	13	Kannst Du erläutern, warum/wie ich die fehler beim compilieren sehen kann?
3687	1:32:37	305	22	Wenn da unten bei dateityp java steht Dann denke ich doch, dass das nich not tut. aber dann mal danke T_{303}! werde ich sofort ändern.
3692	1:33:38	305	40	7:ReadAndWrite.6
3693	1:33:38	305	35	7 ← 1
3694	1:33:38	305	36	0 → 1
3706	1:34:18	305	40	8:ReadAndWrite
3710	1:34:21	305	35	1 ← 7
3712	1:34:35	305	40	9:ReadAndWrite
3724	1:35:06	305	40	11:ReadAndWrite

Fortsetzung auf nächster Seite

Tab. E.12: Aktionen von Teilnehmer T_{305} im Kontext von Problem 6 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
3727	1:35:16	305	35	8 ← 1
3783	1:36:48	305	9	Würdest Du bitte sagen welche frage das war? das mit dem compiler wird ja wohl an der dateiendung liegen, also das hat sich erledigt. dafür hab ich jetzt das problem, dass ich nicht weiß, wie ich meine letzte version herholen und neu abspeichern soll ?
3784	1:36:59	305	35	7 ← 8
3788	1:37:00	305	35	8 ← 0
3789	1:37:00	305	35	8 ← 0
3792	1:38:04	305	35	1 ← 8
3793	1:38:24	305	44	1:ReadAndWrite
3796	1:38:46	305	42	1:ReadAndWrite.java

E.3 Ergänzende Beobachtungen (S_{69} - S_{77})

E.3.1 Benutzeraktionen zu Problembeobachtung 7 (S_{71})

Tab. E.13: Kommunikation aller Teilnehmer im Kontext von Problem 7

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
1402	0:52:58	293	27	Ich denke wir müssen auch den methodennamen anpassen, hab ich gemacht aber läuft immer noch nicht .	292
1432	0:53:55	293	12	Kannst Du mir mehr sagen was du vorhin außer den buttonbeschriftungen noch geändert hast ?	292
1465	0:55:00	292	29	Lasst es mich so erklären ich habe sonst nur die zeilen und spalten anzahl geändert, aber ich glaube, bei der beschriftung ist irgendwas falsch .	293
1520	0:55:49	293	31	Ich bin ziemlich sicher das in den eckigen klammern bei den buttons 5 stehen bleiben muss, weil es ja 5+'0' buttons sind .	292
1524	0:56:21	292	4	Ja .	293

Fortsetzung auf nächster Seite

Tab. E.13: Kommunikation aller Teilnehmer im Kontext von Problem 7 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
1565	0:58:09	293	6	Verzeihung aber muss zwischen den buttonbeschriftungen nicht ein + stehen? .	292
⋮					
Problem 8					
⋮					
2087	1:16:53	292	27	Ich denke ich weiß, wie das mit den button geht .	293
⋮					
2295	1:19:02	293	14	Warum denkst Du, dass in die eckigen klammern jedes mal ein i muss, da gehört doch jeweils die entsprechende zahl hin, oder ?	292
⋮					
2492	1:22:43	293	14	Warum denkst Du, dass diese sachen in die for schleife gehören ?	292
2495	1:24:01	293	31	Ich bin ziemlich sicher das die jeweiligen knöpfe nur einmal einer schrift zugeprdnet werden müssen .	292
⋮					
2499	1:24:22	292	30	Zur Rechtfertigung weiß ich nict genau, ich habs einfach mal ausprobiert .	293
2505	1:25:29	292	6	Verzeihung ich gebs dir mal frei .	293
2508	1:25:37	293	24	Ich bin nicht sicher aber man kann jetzt ger nicht mehr ausführen .	292
2510	1:26:05	292	30	Zur Rechtfertigung aber im compiler ist es richtig .	293
2511	1:26:30	293	12	Kannst Du mir mehr sagen ob es bei dir jetzt richtig ist ?	292
2512	1:26:37	291	12	Kannst Du mir mehr sagen über die (i+1) hinter den getraenken ?	0
2540	1:27:40	292	29	Lasst es mich so erklären das stand schon so in dem programm, aber ich denke, dass ja immer ein vbutton dazukommen muss .	291

Fortsetzung auf nächster Seite

Tab. E.13: Kommunikation aller Teilnehmer im Kontext von Problem 7 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
2541	1:28:22	293	15	Denkst Du wir sollten das überhaupt mit einer schleife machen, eigentlich ist doch jeder button anders und könnte auch einzeln erstellt werden ?	292
2553	1:29:39	292	2	Zusammenfassend naja, die schleife war doch aber schon vorher da .	293
2554	1:30:09	291	6	Verzeihung aber ich weiß warum die array länge fängt doch immer bei null an ein button null wäre aber kein button also muss man immer einen dazu zählen!!!!haha .	0
2555	1:30:20	293	2	Zusammenfassend aber unser beispiel ist doch in bezug auf die buttonbeschriftung ganz anders .	292
2556	1:30:53	292	2	Zusammenfassend dann nimm mal die schleife weg .	293

E.3.2 Benutzeraktionen zu Problembeobachtung 8 (S_{71})

Tab. E.14: Kommunikation aller Teilnehmer im Kontext von Problem 8

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
1631	0:59:22	292	11	Weisst Du was ich bei klasse eingeben muss? ?	293
1632	0:59:52	293	12	Kannst Du mir mehr sagen was du gerade meinst ?	0
1752	1:01:07	292	2	Zusammenfassend also, wenn man es ausführen will, kommt doch ein fenster, und er zeigt als fehler an, dass er das dokument in klasse test 23 nicht findet, aber was muss man sonst eingeben?	293
1773	1:01:30	293	5	Nein .	292
1787	1:01:53	293	29	Lasst es mich so erklären da muss doch der name unserer klasse rein!!! .	292
1839	1:02:20	292	30	Zur Rechtfertigung dann mach mal .	293

Fortsetzung auf nächster Seite

Tab. E.14: Kommunikation aller Teilnehmer im Kontext von Problem 8 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
1890	1:02:59	293	30	Zur Rechtfertigung das hab ich doch schon .	292
1908	1:03:38	292	2	Zusammenfassend und, funktioniert es? .	293
1952	1:04:18	293	29	Lasst es mich so erklären es funktioniert nicht, aber das ist doch die ganze zeit das problem oder nicht? .	292
				⋮	
2009	1:05:57	293	13	Kannst Du erläutern, warum/wie das erste beispiel jetzt auch nicht mehr ausgeführt wird ?	292
2010	1:06:04	292	6	Verzeihung aber kannst du uns sagen, was wir in die felder eingeben müssen, die sich öffnen, wenn wir das programm ausführen wollen? Laut dem compiler ist das programm ok, aber es lässt sich nicht ausführen! .	284
2017	1:06:50	284	6	Verzeihung einen Moment ich probiers eben aus .	292
				⋮	
2051	1:11:37	284	29	Lasst es mich so erklären zum ausfähren musst du den KLassennamen angeben, wenn es trotzdem nicht funktionieren sollte, dann musst du in der Shell auf x klicken um alte Berechnungen zu stoppen .	292
2055	1:12:24	293	28	Zur Ausarbeitung es läuft wenigstens schonmal .	292
2057	1:13:32	292	29	Lasst es mich so erklären bei mir läuft noch gar nichts. sag mir bitte mal genau welchen klassennamen wir haben .	293
2059	1:14:18	293	29	Lasst es mich so erklären du musst AutomatFrame ohne.java eingeben! .	0
2066	1:14:49	284	2	Zusammenfassend : hat es geklappt mit dem Ausführen? Wenn du die Shell beendet hast, musst du die Klasse nochmal ausführen .	292

Fortsetzung auf nächster Seite

Tab. E.14: Kommunikation aller Teilnehmer im Kontext von Problem 8 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
2076	1:15:39	292	4	Ja .	0

E.3.3 Benutzeraktionen zu Problembeobachtung 9 (S_{73})

Tab. E.15: Kommunikation aller Teilnehmer im Kontext von Problem 9

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
3889	0:38:18	312	6	Verzeihung Hast Du Das Programm schon mal laufen lassen, weiß gerade nicht den Ausführungsbefehl .	311
3897	0:39:14	311	27	Ich denke ..... probier es doch in der shell mit java Dateiname .	0
3911	0:39:34	312	4	Ja .	0
3951	0:42:59	311	27	Ich denke , wir sollten mal versuchen, die Klassen mit .java am Ende zu speicher, dann klappt das auch ;) .	0
3992	0:44:32	312	11	Weisst Du welche Parameter ich zur Programmeingabe beim Ausführen eingeben muss ?	306
3998	0:44:50	311	27	Ich denke keine .	0

Tab. E.16: Aktionen von Teilnehmer T_{312} im Kontext von Problem 9

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1422	0:13:30	312	47	1:Getraenke
1424	0:13:44	312	47	1:Getraenke
1427	0:13:53	312	47	1:Getraenke
1428	0:13:55	312	47	1:Getraenke
1429	0:13:55	312	47	1:Getraenke
1430	0:13:55	312	47	1:Getraenke
1453	0:14:13	312	36	$205 \rightarrow 43$
1454	0:14:13	312	52	$< NULL >$
1483	0:14:20	312	52	$< NULL >$
1484	0:14:21	312	38	1:Getraenke
			$\vdots$	
3812	0:36:46	312	39	4:UNTITLED4
3813	0:36:48	312	35	$4 \leftarrow 1$

Fortsetzung auf nächster Seite

Tab. E.16: Aktionen von Teilnehmer T_{312} im Kontext von Problem 9 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
3814	0:36:53	312	36	0 → 1
3815	0:36:53	312	43	4:UNTITLED4
3816	0:36:53	312	35	3 ← 0
3817	0:36:57	312	35	1 ← 3
3818	0:36:57	312	36	147 → 188
3827	0:37:24	312	36	188 → 63
3889	0:37:51	312	6	Verzeihung Hast Du Das Programm schon mal laufen lassen, weiß gerade nicht den Ausführungsbefehl .
3890	0:37:51	312	36	63 → 53
3891	0:38:18	312	35	3 ← 1
3892	0:38:22	312	35	2 ← 3
3893	0:38:24	312	35	1 ← 2
3894	0:38:24	312	36	53 → 63
3895	0:38:24	312	47	1:Getraenke
3896	0:38:27	312	48	0:null
3898	0:38:51	312	36	63 → 157
3899	0:39:14	312	52	$< NULL >$
3911	0:39:25	312	4	Ja .
3912	0:39:28	312	36	157 → 188
3913	0:39:34	312	52	$< NULL >$
3916	0:39:38	312	41	1:Getraenke
3917	0:39:40	312	36	188 → 180
3918	0:39:46	312	36	180 → 176
3919	0:39:51	312	41	1:Getraenke
3938	0:40:52	312	35	2 ← 1
3939	0:40:59	312	35	1 ← 2
3940	0:41:15	312	36	176 → 190
3941	0:41:15	312	36	190 → 175
3953	0:42:59	312	36	175 → 186
3992	0:44:28	312	11	Weisst Du welche Parameter ich zur Programmeingabe beim Ausführen eingeben muss ?

E.3.4 Benutzeraktionen zu Problembeobachtung 10 (S_{73})

Tab. E.17: Kommunikation aller Teilnehmer im Kontext von Problem 10

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
4102	0:48:37	312	6	Verzeihung Läuft Dein Programm .	311
4113	0:49:16	311	27	Ich denke irgendwas passt ihm in Zeile 64 bei dem e.getSource nicht.... k.a. was .	0
4169	0:50:18	312	7	Ich sehe, was Du sagen willst vielleicht sollte ich mal ein wenig code entsorgen... .	311
5105	1:00:42	311	27	Ich denke so ein Käse, da ist kein Fehler drin.... trotzdem willer nich .	0
5114	1:02:23	312	12	Kannst Du mir mehr sagen ob Deine Klasse ohne den Aufruf meines Fensters läuft? ?	311
5130	1:03:20	311	6	Verzeihung nein, da sind noch 3 fehler drin, die ich aber nicht finde .	0
5137	1:04:44	311	11	Weisst Du was an Zeile 63 (buttons.length) falsch ist ?	312
5146	1:08:45	312	7	Ich sehe, was Du sagen willst ich glaube da steckt keine Angabe vom Array [ ] mit im Button-Befehl .	311
5293	1:11:50	311	24	Ich bin nicht sicher , aber ich hab mir das hier so vorgestellt. Du bekommst einen String übergeben, d.h. ich bräuchte in deiner Klasse eine Methode, die da heißt "bestelle(String name)".	0
5294	1:12:29	311	27	Ich denke diese Methode sollte dann beinhalten "setTitle(name)" .	0
5296	1:13:44	312	7	Ich sehe, was Du sagen willst , aber normalerweise reicht es diesen String meinem Konstruktor zu übergeben... .	311
5674	1:17:36	311	13	Kannst Du erläutern, warum/wie Zeile 63 (e.getSource) nicht in Ordnung ist ?	306

Tab. E.18: Aktionen von Teilnehmer T_{311} im Kontext von Problem 10

Lfd.Nr.	Zeit	Sender	Code	Inhalt
4002	0:45:14	311	47	2:AutomatFrame.java
4003	0:45:24	311	36	64 → 69
4008	0:45:43	311	36	69 → 26
4010	0:45:49	311	36	26 → 64
4011	0:46:09	311	36	64 → 65
4012	0:46:19	311	47	2:AutomatFrame.java
4051	0:46:55	311	36	65 → 66
4078	0:47:25	311	36	66 → 32
4079	0:47:41	311	36	32 → 36
4087	0:47:51	311	36	36 → 64
4090	0:48:04	311	36	64 → 38
4091	0:48:07	311	38	2:AutomatFrame.java
4092	0:48:20	311	52	$< NULL >$
4093	0:48:21	311	38	2:AutomatFrame.java
4094	0:48:21	311	52	$< NULL >$
4095	0:48:23	311	38	2:AutomatFrame.java
4096	0:48:23	311	52	$< NULL >$
4097	0:48:23	311	41	2:AutomatFrame.java
4098	0:48:23	311	37	2:AutomatFrame.java
4099	0:48:25	311	37	2:AutomatFrame.java
4100	0:48:32	311	41	2:AutomatFrame.java
4101	0:48:32	311	36	38 → 39
4106	0:48:40	311	36	39 → 40
4113	0:48:56	311	27	Ich denke irgendwas passt ihm in Zeile 64 bei dem e.getSource nicht.... k.a. was .
4114	0:48:56	311	36	40 → 71
4118	0:49:31	311	36	71 → 66
4119	0:49:37	311	36	66 → 67
4120	0:49:54	311	37	2:AutomatFrame.java
			⋮	
5379	1:15:17	311	47	2:AutomatFrame.java
5381	1:15:23	311	38	2:AutomatFrame.java
5382	1:15:30	311	52	$< NULL >$
5383	1:15:36	311	36	65 → 69
5384	1:15:36	311	38	2:AutomatFrame.java
5385	1:15:36	311	52	$< NULL >$
5387	1:15:37	311	41	2:AutomatFrame.java
5388	1:15:40	311	47	2:AutomatFrame.java
5389	1:15:46	311	47	2:AutomatFrame.java
5393	1:15:59	311	36	69 → 65

Fortsetzung auf nächster Seite

Tab. E.18: Aktionen von Teilnehmer T_{311} im Kontext von Problem 10 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
5394	1:15:59	311	37	2:AutomatFrame.java
5395	1:16:00	311	36	65 → 69
5396	1:16:00	311	37	2:AutomatFrame.java
5397	1:16:02	311	41	2:AutomatFrame.java
5402	1:16:06	311	36	69 → 64
5660	1:16:16	311	36	64 → 113
5674	1:17:26	311	13	Kannst Du erläutern, warum/wie Zeile 63 (e.getSource) nicht in Ordnung ist ?
5677	1:17:45	311	36	113 → 82
5678	1:17:47	311	47	2:AutomatFrame.java
5688	1:18:40	311	45	2:AutomatFrame.java
5689	1:18:44	311	35	6 ← 2
5704	1:20:27	311	35	2 ← 6
5707	1:20:45	311	35	1 ← 2
5709	1:20:59	311	35	5 ← 1
5712	1:21:09	311	35	6 ← 5
5713	1:21:10	311	44	6:SwingFensterButtonEnde.java
5714	1:21:12	311	35	2 ← 6
5722	1:21:45	311	44	2:AutomatFrame.java
5726	1:22:17	311	35	3 ← 2
5727	1:22:20	311	35	2 ← 3
5728	1:22:24	311	35	6 ← 2
5729	1:22:24	311	35	5 ← 6
5730	1:22:32	311	35	3 ← 5
5731	1:22:33	311	35	2 ← 3

E.3.5 Benutzeraktionen zu Problembeobachtung 11 (S_{74})

Tab. E.19: Kommunikation aller Teilnehmer im Kontext von Problem 11

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
1577	0:28:43	294	11	Weisst Du ob man noch irgendwas ändern musste ausser dem Datentypen, um das Array zu sortieren? ?	0
1960	0:33:08	286	27	Ich denke ich hab keine Ahnung .	294
2044	0:33:54	294	2	Zusammenfassend ich auch nicht, in er api steht leider auch nix, zumindest hab ich noch nix gefunden .	0

Fortsetzung auf nächster Seite

Tab. E.19: Kommunikation aller Teilnehmer im Kontext von Problem 11 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
2081	0:34:52	286	23	Aber haste schon mal unter Compareable gesucht? Müsste n Interface sein, also krusiv geschrieben ‘ .	0
2446	0:39:24	286	27	Ich denke danke :o) .	0
2588	0:41:46	286	27	Ich denke ich habe fertig .	0
2610	0:42:20	296	6	Verzeihung soll die sortierte liste auch abgespeichert werden? .	0
2647	0:43:26	286	27	Ich denke du musst noch n Compare schreiben .	294
2675	0:44:02	294	25	Mit anderen Worten is erledigt .	286
2789	0:45:09	286	13	Kannst Du erläutern, warum/wie das mit dem vergleichen zweier Strings so funktioniert??? ?	294
2790	0:45:29	294	2	Zusammenfassend ich denke ja, warum?? .	286
2987	0:47:26	286	27	Ich denke du musst noch die METhode Swap schreiben .	0
3682	0:52:18	294	6	Verzeihung , und nun testlauf?? .	286
3685	0:52:38	286	27	Ich denke ich korrigiere erstmal meine 10 Fehler ;-) .	0
4459	1:00:52	296	6	Verzeihung lassen strings sich so vergleichen? .	294
4460	1:01:25	286	27	Ich denke ich bin dumm, kann mir mal bitte wer erklären wieso ich das erste mal auf datei zugreifen kann und er beim zweiten mal meint, as Datei evtl. nicht initialisiert wurde? .	306
4461	1:01:26	294	2	Zusammenfassend anscheinend ja nicht, sonst wärs ja kein fehler .	296
4792	1:08:51	296	6	Verzeihung und hast du denn fehler gefunden? .	294
4904	1:11:23	286	27	Ich denke ich würd sagen es funzt soweit alles .	0
4916	1:12:07	286	27	Ich denke oder auch nicht, die Sortnamen is ja noch leer :-(.	0
5534	1:18:54	286	27	Ich denke wie wärs wenn du nur das erste Char im Wort miteiander vergleichst? .	0
5537	1:19:35	286	27	Ich denke String.charAt(0) .	0

Fortsetzung auf nächster Seite

Tab. E.19: Kommunikation aller Teilnehmer im Kontext von Problem 11 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
5565	1:20:42	286	26	Ich denke, wir sollten du kannst n char in nen int umwandeln .	0
5645	1:22:19	294	2	Zusammenfassend versteh ich nicht so ganz, habs mal freigegeben .	286
5739	1:23:33	286	27	Ich denke so könnts gehen .	0
5799	1:24:08	294	2	Zusammenfassend und der weiß jetzt selbst, das a=1, b=2,...oder wie läuft das? .	286
5801	1:24:59	286	2	Zusammenfassend ne a = 65 b = 66 usw halt Uniciode/ASCII Tabelle .	0
5807	1:25:24	294	0	Okay, lasst uns fortfahren .	286
6226	1:39:32	286	27	Ich denke wenn wir jetzt herraus finden wieso der net abspeichert sind wir fertig .	306

Tab. E.20: Aktionen von Teilnehmer T_{294} im Kontext von Problem 11

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1247	0:24:37	294	35	$3 \leftarrow 4$
1248	0:24:40	294	35	$2 \leftarrow 3$
1249	0:24:42	294	35	$3 \leftarrow 2$
1251	0:24:44	294	35	$5 \leftarrow 3$
1256	0:24:45	294	35	$4 \leftarrow 5$
1268	0:24:53	294	35	$2 \leftarrow 4$
1283	0:24:58	294	35	$4 \leftarrow 2$
1318	0:25:16	294	35	$2 \leftarrow 4$
1319	0:25:26	294	35	$3 \leftarrow 2$
1324	0:25:29	294	35	$2 \leftarrow 3$
1361	0:25:45	294	35	$4 \leftarrow 2$
1411	0:27:34	294	36	$2 \rightarrow 19$
1419	0:27:35	294	36	$19 \rightarrow 16$
1577	0:28:43	294	11	Weisst Du ob man noch irgendwas ändern musste ausser dem Datentypen, um das Array zu sortieren? ?
1593	0:28:51	294	35	$5 \leftarrow 4$
1601	0:29:00	294	35	$4 \leftarrow 5$
1602	0:29:04	294	35	$3 \leftarrow 4$
1603	0:29:06	294	35	$4 \leftarrow 3$
1611	0:29:19	294	36	$16 \rightarrow 13$

Fortsetzung auf nächster Seite

Tab. E.20: Aktionen von Teilnehmer T_{294} im Kontext von Problem 11 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
		⋮		
1676	0:29:35	294	36	20 → 17
2044	0:33:54	294	2	Zusammenfassend ich auch nicht, in er api steht leider auch nix, zumindest hab ich noch nix gefunden .
2460	0:39:49	294	35	3 ← 4
2549	0:40:49	294	35	2 ← 3
2556	0:40:58	294	35	4 ← 2
2572	0:41:05	294	35	2 ← 4
2573	0:41:15	294	35	4 ← 2
2574	0:41:22	294	35	3 ← 4
2575	0:41:24	294	35	4 ← 3
2576	0:41:26	294	36	17 → 4
2577	0:41:28	294	37	4:Sort.java

E.3.6 Benutzeraktionen zu Problembeobachtung 12 (S_{75})

Tab. E.21: Kommunikation aller Teilnehmer im Kontext von Problem 12

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
786	0:23:25	292	2	Zusammenfassend ich habe nicht worklich ahnung, wie so ein sortieralgorithmus aufgebaut ist, oder wie man ihn programmiert. Du? .	293
829	0:25:23	293	28	Zur Ausarbeitung ne, sowas brauchten wir ja bisher auch noch nicht so . es müsste doch aber in den sachen zur letzten vorlesung ein beispiel geben, oder nicht? .	0
830	0:25:46	293	27	Ich denke ansonsten solltest du mal den tutor fragen .	0
994	0:30:15	292	11	Weisst Du wie der sortieralgorithmus fiunktioniert, oder kannst du mir ein beispiel geben? ?	306
999	0:30:40	293	12	Kannst Du mir mehr sagen über den grundsätzlichen aufbau der klasse ReadAndWrite ?	306

Tab. E.22: Aktionen von Teilnehmer T_{292} im Kontext von Problem 12

Lfd.Nr.	Zeit	Sender	Code	Inhalt
786	0:23:25	292	2	Zusammenfassend ich habe nicht worklich ahnung, wie so ein sortieralgorithmus aufgebaut ist, oder wie man ihn programmiert. Du? .
790	0:23:35	292	35	$5 \leftarrow 1$
794	0:23:38	292	35	$2 \leftarrow 5$
805	0:23:52	292	35	$3 \leftarrow 2$
806	0:23:53	292	35	$1 \leftarrow 3$
807	0:23:58	292	37	1:Sort.java
808	0:23:58	292	37	1:Sort.java
809	0:23:58	292	37	1:Sort.java
810	0:23:59	292	37	1:Sort.java
811	0:24:11	292	35	$5 \leftarrow 1$
812	0:24:13	292	35	$2 \leftarrow 5$
813	0:24:38	292	35	$3 \leftarrow 2$
814	0:24:40	292	35	$2 \leftarrow 3$
815	0:24:45	292	35	$1 \leftarrow 2$
816	0:24:48	292	37	1:Sort.java
817	0:24:52	292	35	$2 \leftarrow 1$
818	0:25:06	292	35	$1 \leftarrow 2$
819	0:25:07	292	38	1:Sort.java
820	0:25:07	292	52	$< NULL >$
821	0:25:08	292	38	1:Sort.java
822	0:25:08	292	52	$< NULL >$
823	0:25:08	292	38	1:Sort.java
824	0:25:08	292	52	$< NULL >$
825	0:25:08	292	38	1:Sort.java
826	0:25:08	292	52	$< NULL >$
827	0:25:08	292	38	1:Sort.java
828	0:25:08	292	52	$< NULL >$
831	0:27:16	292	40	6:SortAlgorithms.java
832	0:27:16	292	35	$6 \leftarrow 1$
833	0:27:16	292	36	$0 \rightarrow 1$
834	0:27:30	292	35	$1 \leftarrow 6$
835	0:27:39	292	41	1:Sort.java
836	0:27:39	292	43	1:Sort.java
837	0:27:39	292	35	$2 \leftarrow 2$
840	0:27:42	292	35	$6 \leftarrow 2$
879	0:28:19	292	36	$1 \rightarrow 10$
880	0:28:21	292	43	6:SortAlgorithms.java
881	0:28:22	292	35	$5 \leftarrow 0$

Fortsetzung auf nächster Seite

Tab. E.22: Aktionen von Teilnehmer T_{292} im Kontext von Problem 12 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
889	0:28:32	292	40	7:SortAlgorithms.java
890	0:28:32	292	35	$7 \leftarrow 5$
891	0:28:32	292	36	$0 \rightarrow 1$
920	0:28:42	292	36	$1 \rightarrow 13$
921	0:28:42	292	52	$< NULL >$
923	0:28:43	292	38	7:SortAlgorithms.java
924	0:28:43	292	52	$< NULL >$
925	0:28:43	292	36	$13 \rightarrow 4$
926	0:28:43	292	52	$< NULL >$
927	0:28:49	292	36	$4 \rightarrow 5$
928	0:28:49	292	38	7:SortAlgorithms.java
929	0:28:49	292	52	$< NULL >$
930	0:28:49	292	38	7:SortAlgorithms.java
931	0:28:49	292	52	$< NULL >$
932	0:28:49	292	38	7:SortAlgorithms.java
933	0:28:49	292	52	$< NULL >$
934	0:28:50	292	38	7:SortAlgorithms.java
935	0:28:50	292	52	$< NULL >$
936	0:28:50	292	38	7:SortAlgorithms.java
937	0:28:50	292	52	$< NULL >$
938	0:28:50	292	38	7:SortAlgorithms.java
939	0:28:50	292	52	$< NULL >$
940	0:28:50	292	38	7:SortAlgorithms.java
941	0:28:50	292	52	$< NULL >$
942	0:28:50	292	38	7:SortAlgorithms.java
943	0:28:50	292	52	$< NULL >$
944	0:28:52	292	37	7:SortAlgorithms.java
945	0:28:53	292	36	$5 \rightarrow 7$
994	0:30:15	292	11	Weisst Du wie der sortieralgorithmus fiunktioniert, oder kannst du mir ein beispiel geben? ?

E.3.7 Benutzeraktionen zu Problembeobachtung 13 (S_{77})

Tab. E.23: Kommunikation aller Teilnehmer im Kontext von Problem 13

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1	0:02:08	290	31	Ich bin ziemlich sicher dass ich nicht der beste Mann für die erste Klasse bin .

Fortsetzung auf nächster Seite

Tab. E.23: Kommunikation aller Teilnehmer im Kontext von Problem 13 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
2	0:02:45	288	2	Zusammenfassend möchte ich sagen dass ich keine ahnung von streams habe .
3	0:04:32	290	15	Denkst Du , du bist fit mit Streams? ?
4	0:05:01	288	27	Ich denke wir sollten jetzt erstmal die aufgaben verteilen .
6	0:06:36	316	27	Ich denke einigermasse komme ich klar mit strings .
10	0:08:22	290	27	Ich denke Dann sollte sich T_{316} an den Strings versuchen. .
26	0:09:07	316	27	Ich denke wir müssen doch die aufgaben verteilen .
31	0:09:31	290	4	Ja .
79	0:09:53	316	28	Zur Ausarbeitung hat jemand schon die datei runtergeladen .
80	0:10:01	290	5	Nein .
172	0:12:32	290	21	Deshalb T_{316} such dir einfach was aus, ich mach dann was übrig bleibt .
253	0:14:30	288	2	Zusammenfassend habe ich mal einen anfang gemacht und weiß jetzt aber nicht mehr weiter :-) .
256	0:14:54	316	27	Ich denke das schwierigste ist klasse Readand Write, Sortieren kann ich recht schnell machen und main methode ist nur ein paar zeilen .
257	0:15:22	288	2	Zusammenfassend wer traut sich denn die read an write klasse zu? .
258	0:16:00	316	27	Ich denke machen wir zusammen .
261	0:17:08	288	1	Lasst mich Euch zeigen hier findet ihr was: C:\java\javabuch\k100123.html .
262	0:17:27	290	27	Ich denke ich versuch mich mit dem Sortieren. Und ihr macht zusammen Lesen schreiben und ablauf .
263	0:17:32	288	1	Lasst mich Euch zeigen hier steht was für uns: Listing1809.java .
572	0:19:11	288	1	Lasst mich Euch zeigen guckt euch mal meine klasse an... .
610	0:21:15	316	16	Bitte zeige mir wo liegt die datei namen.txt !
624	0:23:06	288	2	Zusammenfassend wie soll es denn jetzt weitergehen? .

Fortsetzung auf nächster Seite

Tab. E.23: Kommunikation aller Teilnehmer im Kontext von Problem 13 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
626	0:24:14	290	1	Lasst mich Euch zeigen in Zeile drei steht die Methode die ihr zum Sortieren braucht. .
641	0:24:42	316	27	Ich denke wir haben die namen eingelest, jetzt müssen wir die sortieren .
644	0:25:03	288	2	Zusammenfassend wo in zeile drei??? .
645	0:25:12	290	27	Ich denke Ich schau mal wie sich das mit der Pfadangabe machen .
651	0:25:42	290	27	Ich denke Mein Quelltext müsst ihr in Ablauf aufrufen .

Tab. E.24: Aktionen von Teilnehmer T_{288} im Kontext von Problem 13

Lfd.Nr.	Zeit	Sender	Code	Inhalt
241	0:13:37	288	37	1:ReadAndWrite.java
242	0:13:38	288	37	1:ReadAndWrite.java
243	0:13:38	288	36	12 → 13
244	0:13:39	288	37	1:ReadAndWrite.java
245	0:13:39	288	37	1:ReadAndWrite.java
246	0:13:39	288	37	1:ReadAndWrite.java
247	0:13:39	288	36	13 → 14
249	0:13:40	288	37	1:ReadAndWrite.java
250	0:13:40	288	37	1:ReadAndWrite.java
251	0:14:05	288	35	3 ← 1
252	0:14:08	288	35	1 ← 3
253	0:14:30	288	2	Zusammenfassend habe ich mal einen anfang gemacht und weiß jetzt aber nicht mehr weiter :-) .
254	0:14:41	288	41	1:ReadAndWrite.java
255	0:14:41	288	45	1:ReadAndWrite.java
			⋮	
608	0:21:11	288	35	2 ← 1
609	0:21:13	288	35	1 ← 2
611	0:21:22	288	35	3 ← 1
613	0:21:24	288	35	1 ← 3
617	0:22:08	288	35	2 ← 1
618	0:22:09	288	35	3 ← 2
619	0:22:10	288	35	1 ← 3

Fortsetzung auf nächster Seite

Tab. E.24: Aktionen von Teilnehmer T_{288} im Kontext von Problem 13 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
624	0:23:06	288	2	Zusammenfassend wie soll es denn jetzt weitergehen? .
638	0:24:25	288	35	2 ← 1
639	0:24:26	288	35	1 ← 2
644	0:25:03	288	2	Zusammenfassend wo in zeile drei??? .
646	0:25:24	288	35	2 ← 1
647	0:25:29	288	35	3 ← 2
649	0:25:36	288	35	2 ← 3
650	0:25:36	288	35	1 ← 2
652	0:25:54	288	35	2 ← 1
654	0:25:59	288	35	1 ← 2
656	0:26:29	288	2	Zusammenfassend wer macht denn jetzt das einlesen der datei zeilenweise in ein string array? .
659	0:28:00	288	2	Zusammenfassend hat also keiner lust das zu übernehmen? ;) .
660	0:28:23	288	35	2 ← 1
661	0:28:24	288	35	3 ← 2
666	0:30:56	288	35	1 ← 3

E.3.8 Benutzeraktionen zu Problembeobachtung 14 (S_{77})

Tab. E.25: Kommunikation aller Teilnehmer im Kontext von Problem 14

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
610	0:21:15	316	16	Bitte zeige mir wo liegt die datei namen.txt!	0
				⋮	
645	0:25:12	290	27	Ich denke Ich schau mal wie sich das mit der Pfadangabe machen.	0
				⋮	
1720	0:49:09	290	13	Kannst Du erläutern, warum/wie ich eine relative Pfadangabe schreibe ?	306

Tab. E.26: Aktionen von Teilnehmer T_{316} im Kontext von Problem 14

Lfd.Nr.	Zeit	Sender	Code	Inhalt
589	0:20:44	316	35	3 ← 1
599	0:20:48	316	35	2 ← 3
610	0:21:15	316	16	Bitte zeige mir wo liegt die datei namen.txt !
612	0:21:23	316	35	1 ← 2
620	0:22:10	316	35	2 ← 1
625	0:23:15	316	35	1 ← 2
629	0:24:21	316	35	2 ← 1
637	0:24:25	316	35	1 ← 2
640	0:24:30	316	35	2 ← 1
641	0:24:42	316	27	Ich denke wir haben die namen eingelest, jetzt müssen wir die sortieren .
642	0:24:43	316	35	1 ← 2
643	0:24:47	316	35	2 ← 1
648	0:25:35	316	35	1 ← 2
653	0:25:57	316	35	2 ← 1
655	0:26:26	316	35	1 ← 2
657	0:26:30	316	35	2 ← 1
658	0:26:37	316	35	1 ← 2

Tab. E.27: Aktionen von Teilnehmer T_{290} im Kontext von Problem 14

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
645	0:25:12	290	27	Ich denke Ich schau mal wie sich das mit der Pfadangabe machen .	0
651	0:25:42	290	27	Ich denke Mein Quelltext müsst ihr in Ablauf aufrufen .	0
667	0:31:17	290	41	2:Sort.java	0
668	0:31:26	290	35	3 ← 2	0
670	0:31:36	290	35	2 ← 3	0
671	0:31:37	290	36	3 → 4	0
672	0:31:39	290	37	2:Sort.java	0
⋮					
1492	0:46:38	290	37	2:Sort.java	0
1493	0:46:39	290	37	2:Sort.java	0
1494	0:46:43	290	41	2:Sort.java	0
1495	0:46:45	290	47	2:Sort.java	0
1504	0:47:28	290	27	Ich denke Sort.java ist fertig .	0

Fortsetzung auf nächster Seite

Tab. E.27: Aktionen von Teilnehmer T_{290} im Kontext von Problem 14 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
1720	0:49:09	290	13	Kannst Du erläutern, warum/wie ich eine relative Pfadangabe schreibe?	306
1778	0:49:29	290	35	5 ← 2	0
1831	0:49:47	290	39	6:UNTITLED6	0
1833	0:49:47	290	35	6 ← 5	0
1834	0:49:48	290	36	0 → 1	0
1873	0:50:10	290	42	6:Sorttest.java	0
1876	0:50:15	290	37	6:Sorttest.java	0

E.3.9 Benutzeraktionen zu Problembeobachtung 15 (S_{77})

Tab. E.28: Kommunikation aller Teilnehmer im Kontext von Problem 15

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
656	0:26:29	288	2	Zusammenfassend wer macht denn jetzt das einlesen der datei zeilenweise in ein string array?	0
659	0:28:00	288	2	Zusammenfassend hat also keiner lust das zu übernehmen? ;) .	0
662	0:29:31	316	22	Wenn ich die information darüber gleich finde Dann kann auch ich das übernehmen.	0
669	0:31:28	288	2	Zusammenfassend das ist super! dann mache ich die klasse zum starten und T_{290} die sortierung... .	0
865	0:35:44	316	24	Ich bin nicht sicher wie kann man das ganze in string array packen.	0
				⋮	
1042	0:38:06	288	2	Zusammenfassend kann man in untiteld4 sehen wie zeilenweises einlesen funktionieren soll.	0
1087	0:39:18	316	27	Ich denke Untitled4 das ist gute idee .	0
1088	0:39:23	290	27	Ich denke Du musst zuerst herausfinden wie viele zeilen es gibt, und erstellst dann ein Array entsprechend dieser Größe.	316
				⋮	

Fortsetzung auf nächster Seite

Tab. E.28: Kommunikation aller Teilnehmer im Kontext von Problem 15 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
2177	0:56:18	316	28	Zur Ausarbeitung ich brauche Untitled4.	288
2311	0:59:08	288	2	Zusammenfassend hier noch eine möglichkeit die lnge der datei zu ermitteln: .	0
2315	0:59:13	288	2	Zusammenfassend LineNumberInputStream lsis = new LineNumberInputStream(new FileInputStream(file)); while ((c = lsis.read())!= -1) ; System.out.print(lsis.getLineNumber());	0
2360	1:01:58	316	11	Weisst Du ob es so geht(es ist nicht ganz fertig) wie ich es in ReadandWrite gemacht habe ?	288
2468	1:03:19	288	2	Zusammenfassend : probieren geht über studieren! :) teste es doch mal .	316
2644	1:06:15	290	27	Ich denke Das Sortieren ist in den Ablalf eingebunden .	0
2656	1:09:03	316	24	Ich bin nicht sicher aber ich glaube ReaddanWrite ist fertig. Oder habt ihr welche vorschläge .	0
2657	1:09:14	288	2	Zusammenfassend ok .	290

Tab. E.29: Aktionen von Teilnehmer T_{316} im Kontext von Problem 15

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1087	0:39:18	316	27	Ich denke Untitled4 das ist gute idee .
1094	0:39:39	316	35	$2 \leftarrow 4$
1095	0:39:40	316	35	$4 \leftarrow 2$
1099	0:40:00	316	35	$2 \leftarrow 4$
1100	0:40:22	316	35	$1 \leftarrow 2$
1101	0:40:35	316	35	$4 \leftarrow 1$
1250	0:42:38	316	35	$1 \leftarrow 4$
1254	0:42:41	316	35	$3 \leftarrow 1$
1256	0:42:42	316	35	$2 \leftarrow 3$
1257	0:42:50	316	35	$1 \leftarrow 2$
1258	0:42:53	316	35	$4 \leftarrow 1$

Fortsetzung auf nächster Seite

Tab. E.29: Aktionen von Teilnehmer T_{316} im Kontext von Problem 15 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt
1322	0:44:16	316	35	$1 \leftarrow 4$
1331	0:44:26	316	35	$4 \leftarrow 1$
1342	0:44:34	316	35	$1 \leftarrow 4$
1366	0:44:52	316	44	1:ReadAndWrite.java
1369	0:44:53	316	36	$14 \rightarrow 6$
				$\vdots$
1957	0:51:37	316	37	1:ReadAndWrite.java
1959	0:51:50	316	35	$8 \leftarrow 1$
1966	0:52:01	316	35	$1 \leftarrow 8$
1971	0:52:07	316	35	$7 \leftarrow 1$
1977	0:52:21	316	35	$3 \leftarrow 7$
1978	0:52:23	316	35	$4 \leftarrow 3$
1997	0:52:47	316	35	$1 \leftarrow 4$
2014	0:53:02	316	35	$4 \leftarrow 1$
2028	0:53:40	316	44	4:UNTITLED4
2085	0:54:31	316	35	$1 \leftarrow 4$
2088	0:54:32	316	36	$10 \rightarrow 7$
2134	0:54:56	316	35	$4 \leftarrow 1$
2137	0:55:02	316	35	$1 \leftarrow 4$
2139	0:55:05	316	36	$7 \rightarrow 10$
2140	0:55:09	316	35	$4 \leftarrow 1$
2175	0:55:55	316	35	$1 \leftarrow 4$
2177	0:56:18	316	28	Zur Ausarbeitung ich brauche Untitled4 .

E.4 Beispiel eines Konflikts

In allen durchgeführten Sitzungen gab es bislang nur eine einzige Gruppe[1], in der – verursacht durch persönliche Animositäten zwischen einzelnen Gruppenmitgliedern – wiederholt Konflikte auftraten, die einen reibungslosen Ablauf der Aufgabenbearbeitung teils massiv störten. Die folgende Tabelle E.30 enthält einen Auszug aus der ersten Sitzung der Gruppe G_9, die sich auf die Kommunikationsbeiträge der Teammitglieder beschränkt, das heißt sonstige Aktionen der Teilnehmer (wie bspw. Dokumentenbearbeitung oder Navigation im Gruppeneditor) wurden aus Gründen der Übersichtlichkeit aus dem Protokoll herausgefiltert.

Die nachfolgend dargelegte Kommunikation der Sitzung von Gruppe G_9 beginnt mit einer (hier nicht wiedergegebenen) Phase der Arbeitsvorbereitung, die auch

[1] Diese Gruppe wird im Folgenden G_9 genannt. Sie besteht aus den Teilnehmern T_{29}, T_{30} und T_{31} sowie dem Tutor T_1

eine Aufgabenverteilung umfasst. Es folgt eine kurze Phase der Aufgabenbearbeitung, bestehend aus diversen Dokumentenoperationen der Teilnehmer, bevor der für den Konflikt relevanten Teil der Sitzung beginnt.

Tab. E.30: Konflikt am Beispiel von Benutzertest S_7

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
23	0:35:21	29	27	Ich denke T_{31} Aufg. 1, T_{29} Aufg. 2, T_{30} Aufg. 3! .	
425	0:38:50	29	12	Kannst Du mir mehr sagen über deine vorgehensweise? ?	
536	0:39:34	29	12	Kannst Du mir mehr sagen T_{30}? du machst die main-methode... ?	30
547	0:39:56	29	9	Würdest Du bitte dein dokument freigeben? ?	30
589	0:40:34	30	25	Mit anderen Worten o.k., .	29
610	0:42:38	30	28	Ich bin ziemlich sicher Wofür bracust Du den 2-Zeiler?, mache weiter .	29
627	0:43:44	29	11	Weisst Du nach was für Kriterien die Dualzahlen sortiert werden? ich mein den technischen vorgang?? ?	
629	0:44:20	30	2	Zusammenfassend wenn ihr d. methoden schreibt, erstelle ich nur dir MAIN? .	
630	0:44:30	29	4	Ja .	
675	0:45:49	29	22	Wenn der computer die zahlen sortiertDann macht er das WIE??.	
685	0:47:38	30	29	Lasst es mich so erklären Freigeben– anklicken– selber Button zum freigen, fertig .	29
806	0:51:48	31	11	Weisst Du wie ich bei Modulo nicht nur den Rest, sondern auch die Anzahl des Moduls in einer Zahl feststellen kann? Beispiel: 333 % 2 ist Rest 1, aber wie oft ist die 2 drinnen?? Wie mach ich das am besten ?	
878	0:52:58	29	6	Verzeihung aber keine ahnung .	

Fortsetzung auf nächster Seite

Tab. E.30: Konflikt am Beispiel von Benutzertest S_7 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
946	0:53:50	29	27	Ich denke T_1 hat recht damit, dass ein wildes durcheinander programmieren ohne sinn und verstand ist, weil die variablen schon einheitliche namen tragen müssen... wie sollen "die" sich nachher finden?? .	
949	0:56:09	31	18	Ich stimme zu, weil bla. Denke ihr beiden müsst euch am meisten koordinieren. ich gebe ja nur Strings mit Tastatur.leseDualzahl() aus .... irgendwann mal ... hoffentlich ;-)	
1050	0:57:19	29	27	Ich denke carsten, dass du dich wie immer nicht integrierst! .	
1114	0:58:38	31	7	Ich sehe, was Du sagen willst . Was soll ich deiner Meinung nach tun? meine Methode hat nichts mit euren zu tun, da kann ich nichts machen, sorry .	
1136	0:59:49	29	23	Aber als Teil des Teams kannst Du trotzdem zu einer Allgemeinlösung beitragen. Das setzt z.B. ein Vorbereiten ALLER voraus!!! .	
1196	1:00:32	30	25	Mit anderen Worten Ich erstelle ne Methode zum Aufruf eurer methoden, damit bin ich doch sofort fertig o. ? .	
1206	1:00:42	31	25	Mit anderen Worten soll ich jetzt eure Parameter und Methodennamen unter einen Hut bringen?? .	
1229	1:01:11	31	27	Ich denke mal ja .	
1274	1:02:18	29	19	Ich stimme nicht zu, weil du verstehst den kern der sache nicht. also nochmal: bevor man eine aufgabe bearbeitet, sollte man sich kurtzschließen.. T_{30} z.b. hält sich von jeglicher kommunikation fern.. so kann ich auf jeden fall nicht arbeiten! .	
1281	1:02:35	30	28	Ich denke, wir sollten wie können wir uns abstimmen, kann ich unterstuetzen .	29

Fortsetzung auf nächster Seite

Tab. E.30: Konflikt am Beispiel von Benutzertest S_7 *Forts.*

Lfd.Nr.	Zeit	Sender	Code	Inhalt	Empf.
1282	1:02:48	29	11	Weisst Du es geht hier auch nicht darum, dass DU was tust, sondern dass WIR was tun.. ?	31
1283	1:03:08	29	27	Ich denke also konstruktive Vorschläge wären angebracht! .	31
1677	1:10:55	30	2	Zusammenfassend Fertig? .	31
1681	1:11:42	31	5	Nein .	
1889	1:14:05	30	6	Verzeihung machst Du jetzt Teil 3 .	29
2221	1:16:42	29	29	Lasst es mich so erklären : ich versuche es.... .	
2255	1:17:19	29	21	Deshalb weil, ich erstmal einen Überblick brauche.. aber lass dich nicht stören ;-)	30
2698	1:11:40	31	11	Weisst Du ob ich das so mit dem temp = (int) x machen kann?? ?	30
2830	1:22:28	31	2	Zusammenfassend . T_{29} ich hab meinen Methodennamen deinem Aufruf angepasst. Wenn ich die in Tastatur.java einfüge findet dein Prog die dann auch. Toll oder? .	
2985	1:24:32	30	6	Verzeihung ich finde im skript nichts dazu .	31
3050	1:25:48	31	24	Ich bin nicht sicher , aber ich wollte nur den wert des vielfachen von 2 haben. der nachkomma anteil ist ja immer der rest beim modulo. verstehst du was ich meine? ist das ding, was ich vorhin schon gefragt hatte .	
3232	1:27:41	30	7	Ich sehe, was Du sagen willst , das Vorgehen ist mir klar, die Umsetzung..../Anweisungen/? .	31
3344	1:28:49	31	1	Lasst mich Euch zeigen was ich meine. (siehe ende Tastatur.java) .	

Zunächst fragt Teilnehmer T_{29} bei T_{30} nach dessen Vorgehensweise nach (t_{425}) und wiederholt diese Frage (t_{536}), als eine Antwort von T_{30} ausbleibt, um kurz darauf dessen aktuelles Dokument anzufordern (t_{547}), was dann auch tatsächlich von T_{30} bestätigt wird (t_{589}). Anschließend stellen sowohl T_{30} als auch T_{29} unterschiedliche Fragen (t_{610} und t_{627}), woraufhin die Kommunikation zunächst einen thematisch inkoheränten Verlauf annimmt (vgl. GÖLDNER, 2005, S56f). Kurze Zeit später meldet T_{31} seinen Teammitgliedern ein fachliches Problem (t_{806}), erhält jedoch durch das Team keine konstruktiven Lösungsvorschläge.

Diese Gesamtsituation greift T_{29} auf, um über die allgemeine Vorgehensweise der Gruppe zu reflektieren (t_{946}). T_{31} stimmt der Kritik von T_{29} zu (t_{949}), weist eine Verantwortung für ein unkoordiniertes und erfolgloses Vorgehen jedoch den beiden anderen Teammitgliedern zu, woraufhin T_{31} von T_{29} persönlich angegriffen wird (t_{1050}). Spätestens mit diesem Beitrag ist der persönliche Konflikt zwischen den Teilnehmern T_{29} und T_{31} offensichtlich, in dessen Verlauf T_{31} von T_{29} mit weiteren Anschuldigungen konfrontiert wird (t_{1136}, t_{1274} und t_{1282}), um dann letztlich zu konstruktiven Vorschlägen aufzufordern (t_{1283}). Der aufgetretene Konflikt kann mit dieser Äußerung als beendet betrachtet werden, da im Anschluss eine Phase der Aufgabenbearbeitung durch alle Mitglieder der Gruppe erfolgt, die sich bis zum Ende der Sitzung erstreckte.

Eine spätere Sitzung derselben Gruppe gipfelte bereits nach einer kurzen Phase erfolgloser Aufgabenverteilung in einem weiteren Konflikt, dessen trauriger Höhepunkt der nachfolgenden Tabelle E.31 entnommen werden kann.

Tab. E.31: Konflikt am Beispiel von Benutzertest S_{13}

Lfd.Nr.	Zeit	Sender	Code	Inhalt
82	0:19:17	29	28	Zur Ausarbeitung : sag mal T_{31}: tut mir leid dass ich schon wieder persönlich werde, aber das bist du auch die ganze zeit:: weißt du was team work iost und wie man wissenschaftlich arbeitet?? und spiel dich nicht so auf: wenn man es genau betrachtet, hast du für jede java-hausarbeit am wenigsten gemacht.. was ja nun nicht gerade für deine hier vorgespielte programmierfähigkeit spricht! .
83	0:20:45	31	22	Wenn du jetzt so anfängst Dann steig ich gleich aus. Erst sollte ich hier die Aufgaben definieren und ne Reihenfolge festlegen und dann bin ich dabei und DU fängst hier an rumzupissen. Hallo ??? Gehts noch????.

Interessant an diesen beiden Beispielen ist nicht nur der Konflikt zwischen T_{29} und T_{31} als solcher, sondern darüber hinaus auch die Tatsache, dass er parallel zur eigentlichen Aufgabenbearbeitung stattfindet: Teammitglied T_{30} hält sich nicht nur so weit wie möglich aus dem Konflikt heraus, sondern versucht währenddessen, den fachlichen Prozess fortzuführen (s. t_{1196} und t_{1281} in Tab.E.30).

Glossar

A

API *Application Programming Interface*
(dt.: Anwendungsprogrammierschnittstelle)
Ein API spezifiziert die Schnittstelle zwischen dem Benutzercode und einem vorgegebenen Programmiersystem wie beispielsweise einem Betriebssytem oder einer Java-Klassenbibliothek. Der Funktionsumfang der Java-Laufzeitbibliothek wird als Java-API bezeichnet und umfasst Beschreibungen darüber, wie die einzelnen Klassen dieser Bibliothek genutzt werden können.
Literatur: Broy & Spaniol (1999), S.28; Schiedermeier (2005), S.46; Savitch (2007), S.197

AWT *Abstract Window Toolkit*
(dt.: Abstrakter Fenster-Werkzeugsatz)
Das AWT ist eine Java-Klassenbibiliothek und stellt eine Standard-API zum Erstellen von plattformunabhängigen fensterbasierten GUI-Anwendungen dar.
Literatur: Campione & Walrath (1998a), S.425ff; Liang (2005), S.371ff
Homepage: `http://java.sun.com/products/jdk/awt/` (Letzter Zugriff: 25.07.2007)

C

CBR *Case Based Reasoning*
(dt.: Fallbasiertes Schließen)
Hierbei handelt es such um ein maschinelles Lernverfahren aus dem Gebiet der KI, das auf bereits vorhandenem Vorwissen basiert. Literatur: Dorn & Gottlob (1999), S.992f; Russell & Norvig (2004), S.864

CBT *Computer Based Training*
(dt.: Computer-basiertes Training)
Dieser Begriff umfasst sämtliche Lernformen, die auf der Verwendung von Computern und dem Einsatz der sogenannten *Neuen*

Medien (auch: digitale Medien) basieren. CBT wird häufig synonym mit E-Learning verwendet.
Literatur: Kerres (2001), S.13f; Lang (2002), S.37 ; Dittler (2003b), S.23ff

CLS *Collaborative Learning Skills*
Eine Taxonomie kommunikativer Akte, entwickelt von McManus & Aiken (1995) im Rahmen ihrer Untersuchungen bzgl. der Zusammenarbeit virtueller Teams bei der Bewältigung von Problemsituationen. Dieser Ansatz wurde in den Arbeiten von Soller (2004) und an den Kontext der objektorientierten Modellierung mittels OMT adaptiert. Die resultierende Codierung von Kommunikation ermöglicht eine relativ einfache Analyse der Zusammenarbeit innerhalb virtueller Teams.
Literatur: McManus & Aiken (1995); Soller (2004)

CORBA *Common Object Request Broker Architecture*
Bei CORBA handelst es sich um eine objektorientierte Middleware, die plattformübergreifende Protokolle und Dienste definiert und von der Object Management Group (OMG) entwickelt wird.
Literatur: Boger (1999), S.85ff; Sommerville (2007), S.278ff
Homepage: (Letzter Zugriff: 25.07.2007)

CSCL *Computer Supported Cooperative Learning*
(dt.: Computer-unterstütztes kooperatives Lernen)
CSCL lässt sich als eine Lernform definieren, bei der mehrere Personen unter nicht ausschließlicher Nutzung von Computern und Computernetzen kooperativ Wissen austauschen und aufbauen.
Literatur: Haake et al. (2004); Wessner (2001)

CSCW *Computer Supported Cooperative Work*
(dt.: Computer-unterstütztes kooperatives Arbeit)
Innerhalb dieses Forschungsgebiets wird auf interdisziplinärer Basis untersucht, wie Individuen in Arbeitsgruppen oder Teams zusammenarbeiten und wie sie dabei durch Informations- und Kommunikationstechnologie unterstützt werden können.
Literatur: Teufel et al. (1995); Döring (2003), S.284ff

cvK *Computer-vermittelte Kommunikation*
Die cvK umfasst alle Arten der Kommunikation, die mittels Unterstützung durch miteinander vernetzte Rechnersysteme zustande kommt.
Synonym: CMC (Computer Mediated Communication)
Literatur: Döring (2003), S.43ff; Schümmer & Haake (2004)

CVS *Concurrent Versions System*
(dt.: nebenläufiges Versionssystem)

CVS ist ein Software-System zur Versionsverwaltung von Dateien. Es wird hauptsächlich für die Verwaltung von Software-Quelltexten verwendet.
Homepage: `http://www.nongnu.org/cvs/`

D

DBMS *Database Management System*
(dt.: Datenbankverwaltungssystem)
Unter einem DBMS versteht man eine Ansammlung mehrerer Programme, mit deren Hilfe ein Anwender eine Datenbank erstellen und pflegen kann. Literatur: Elmasri & Navathe (2005), S.19; Broy & Spaniol (1999), S.165

DEA *Deterministischer Endlicher Automat*
Ein DEA (auch EA genannt) ist ein Konstrukt der Theoretischen Informatik, mit dessen Hilfe eine Maschine modelliert werden kann, die (zeichenweise) eine Eingabe verarbeitet, dabei einen Zustandswechsel durchführt und gegebenenfalls eine Ausgabe erzeugt.
Synonyme: EA (Endlicher Automat), DFA (Deterministic Finite Automaton)
Literatur: Hopcroft et al. (2002), S.45ff; Kastens & Kleine Büning (2005), S.196ff;

DNS *Domain Name Service*
(dt.: Domänennamensdienst)
Dieser Dienst ist zuständig für die Zuordnung der Rechnernamen zu ihren Internet-Adressen und umgekehrt.
Literatur: Tanenbaum (2003a), S.631ff; Kurose & Ross (2005), S.123ff

DOM *Document Object Model*
Bei DOM handelt es sich um ein plattformunabhängiges API, das den Zugriff auf HTML- und XML-Dokumente spezifiziert. Die Pflege wird durch das W3C geregelt.
Literatur: Hégaret et al. (2005); McLaughlin (2001), S.184ff
Homepage: (Letzter Zugriff: 25.07.2007)

F

FIFO *First In, First Out*
In der Informatik wird damit das Arbeitsprinzip einer Warteschlange (engl. *queue*) bezeichnet: Neue Elemente werden an das Ende einer Warteschlange angehängt und vom Kopf derselben

entfernt, so dass jeweils das am längsten wartende Element als nächstes verarbeitet wird. Warteschlangen werden unter anderem dazu verwendet, um den Datenaustausch zwischen zwei Komponenten (Sender und Empfänger) zeitlich zu entkoppeln.
Literatur: Rembold & Levi (1999), S.166f; Broy & Spaniol (1999), S.272; Balzert (1999), S.422

FTP *File Transfer Protocol*
(dt.: Dateiübertragungsprotokoll)
Hierbei handelt es sich um ein Protokoll, das den Austausch von Dateien zwischen entfernten Rechnern regelt.
Literatur: Kurose & Ross (2005), S.166ff; Tanenbaum (2003a), S.677f; Fuhrberg (2000), S.30ff;

G

GUI *Graphical User Interface*
(dt.: Graphische Benutzerschnittstelle)
Hierunter versteht man gemeinhin einen Graphikbildschirm, bestehend aus einer Arbeitsoberfläche, Fenstern, Graphiksymbolen und verschiedenen anderen Bedienelementen, über die der Benutzer mit der Anwendungssoftware interagiert und kommuniziert.
Synonym: Graphische Benutzungsoberfläche
Literatur: Tanenbaum (2003b), S.27f; Balzert (1999), S.30;

H

HTML *Hypertext Markup Language*
(dt.: Hypertext-Beschreibungssprache)
HTML ist eine vom W3C standardisierte Sprache zur Beschreibung von Web-Seiten.
Literatur: Steffen (1998); Dellwig (2000); Tanenbaum (2003a), S.668ff

HTTP *Hypertext Transfer Protocol*
HTTP gilt derzeit als das Standardübertragungsprotokoll im Internet. Es wird hauptsächlich eingesetzt, um Webseiten und andere Daten aus dem WWW in einen Webbrowser zu laden.
Literatur: Kurose & Ross (2005), S.87ff; Tanenbaum (2003a), S.707f; Fuhrberg (2000), S.34f; Döring (2003), S.8ff; Teufel et al. (1995), S.165ff

HTTPS *Secure Hypertext Transfer Protocol*
Hierbei handelt es sich um die mit SSL abgesicherte Variante von

HTTP.
Literatur: Kurose & Ross (2005), S.708ff; Tanenbaum (2003a), S.876ff

I

ICLS *Intelligent Cooperative/Collaborative Learning System*
(dt.: Intelligentes kooperatives Lernsystem)
Unter ICLS, einer Synthese aus CSCL und ITS, versteht man eine intelligente Lernumgebung, die das gemeinsame Lernen innerhalb virtueller Teams unterstützt und eine tutorielle Komponente besitzt.
Literatur: Soller (2001), S.41; McManus & Aiken (1995), S.307

IDE *Integrated Development Environment*
(dt.: Integrierte Entwicklungsumgebung)
Eine IDE ist eine Applikation, die innerhalb einer (meist grapisch basierten) Oberfläche alle Werkzeuge zusammenfasst, die für die Erstellung von Programmen (in einer von der IDE unterstützten Programmiersprache) notwendig sind.
Literatur: Floyd & Züllighoven (1999), S.787; Zuser et al. (2004), S.284f

IIM *Internationales Informationsmanagement*
Hierbei handelt es sich um einen Magister-Studiengang an der Universität Hildesheim, dessen Studierende im Bereich der Informationswissenschaft auch an die Programmierung in Java herangeführt werden.

IMAP *Internet Message Access Protocol*
Dieses Protokoll realisiert die Verwaltung von elektronischen Nachrichten (E-Mails) auf einem Server, ohne dass die Nachricht auf den Client geladen werden muss.
Literatur: Kurose & Ross (2005), S.119ff; Tanenbaum (2003a), S.661f; Fuhrberg (2000), S.439

IPA *Interaction Process Analysis*
(dt.: Interaktions-Prozess-Analyse)
IPA ist ein von Robert Freed Bales entwickeltes formales System zur Messung von Beobachtungen durch Kodierung der Interaktion von Mitgliedern kleiner Gruppen. Die Methode beinhaltet Kategorien und Verfahren zur Kodierung von Interaktion.
Literatur: Bales (1950); Kauffeld (2001), S.54

IRC *Internet Relay Chat*
Dieses Protokoll wird zur Übertragung der Daten bei textbasier-

ten Konferenzen eingesetzt. Es kann als eine der einfachsten Formen von cvK angesehen werden.
Synonym: Chat
Literatur: Fuhrberg (2000), S.35ff; Balzert (1999), S.48

ITS *Intelligentes Tutorielles System*
Ein ITS ist ein E-Learning-System, das in der Lage ist, sich mittels (künstlicher) Intelligenz an den Benutzer anzupassen und diesen in seinem Lernprozess gezielt und angemessen zu unterstützen.
Literatur: Kerres (2001), S.71

J

JDBC *Java Database Connectivity*
Bei JDBC handelt es sich um ein Java-API in Form einer Menge von Schnittstellenklassen, die den Zugriff auf unterschiedlichste (relationale) Datenbanken vereinheitlichen. Es kann somit als eine Art universelle Datenbankschnittstelle aufgefasst werden. Zum Funktionsumfang gehören Aufbau und Verwaltung von Datenbenkverbindungen, die Übergabe von SQL-Befehlen von Java-Programmen an die Datenbank sowie die Bereitstellung und Nutzbarmachung der Ergebnisse in Java-Prorgammen. Der Zugriff auf spezifische Datenbanken wird mittels Treiber realisiert, die die JDBC-Spezifikation umsetzen.
Literatur: Hamilton et al. (1998);
Homepage: `http://java.sun.com/javase/technologies/database/` (Letzter Zugriff: 25.07.2007)

JDK *Java Development Kit*
(dt.: Java-Entwicklungssystem)
Ein JDK enthält alle Komponenten und Werkzeug, die für die Ausführung und die Entwicklung von Java-Prgrammen notwendig sind. Neben der Laufzeitumgebung (s. JRE) und den Standard-Klassenbibliotheken gehören dazu beispielsweise auch der Java-Compiler sowie die Dokumentation der Klassenbibliotheken.
Literatur: Schiedermeier (2005), S.10f; Deitel (2005), S.2f; Bishop (2000), S.49ff
Homepage: `http://java.sun.com/` (Letzter Zugriff: 25.07.2007)

JFC *Java Foundation Classes*
(dt.: Java-Basis-Klassen)
Die JFC wurden mit der Java-Version 1.2 eingeführt und enthalten einen Vielzahl von Klassen, die diverse Vereinfachungen und Verbesserungen beim Erstellen von GUI-Anwendungen (Swing-Klassen), beim Arbeiten mit Container-Klassen (Collection-Klassen) oder auch beim Zugriff auf Klassen- und Objektinfor-

mationen zur Laufzeit (Reflection-API).
Literatur: Campione & Walrath (1998b)

JRE *Java Runtime Environment*
(dt.: Java-Laufzeitumgebung)
Ein JRE beinhaltet alle Komponenten, die für die Ausführung von Java-Programmen, die bereits in Form von Java-Bytecode vorliegen, notwendig sind. Im Gegensatz zu einem JDK fehlen Werkzeuge für die Entwicklung von Java-Programmen, wie beispielsweise der Compiler.
Literatur: Schiedermeier (2005), S.12; Liang (2005), S.532
Homepage: `http://java.sun.com/` (Letzter Zugriff: 25.07.2007)

K

KI *Künstliche Intelligenz*
Dieses Teilgebiet der Informatik befasst sich mit Konzepten, menschliche Intelligenz auf Rechensystemen nachzubilden, sowie mit Algorithmen für intelligente Problemlösungen.
Synonym: AI (Artificial Intelligence)
Literatur: Russell & Norvig (2004); Broy & Spaniol (1999), S.387

KONFKOD *Konferenz-Kodierung*
Dieses prozessanalytische Verfahren der Teamdiagnose von R. Fisch (1994) beruht auf IPA und kodiert Äusserungen von Gruppenmitgliedern in die Kategorien *Lenkung der Diskussion*, *aufgabenbezogene Beiträge* und *sozio-emotionale Beiträge*.
Literatur: Kauffeld (2001), S.55; Hupfeld-Heinemann (2001), S.7

L

LAN *Local Area Network*
(dt.: Lokales Netzwerk)
Ein LAN ist ein Netzwerk, das in seiner Ausbreitung beschränkt ist, dabei aber vergleichsweise hohe Datenraten bereitstellt. In diese Kategorie fallen üblicherweise Installationen innerhalb von Gebäuden oder Gebäudekomplexen, von der Privatwohnung bis hin zur Industrieanlage eines Unternehmens an einem Standort.
Literatur: Plattner & Schulthess (1999), S.383; Broy & Spaniol (1999), S.393; Tanenbaum (2003a), S.31

N

NNTP *Network News Transfer Protocol*
Hierbei handelt es sich um das Standardprotokoll zum Übertragen von Newsartikeln im Internet.
Literatur: Fuhrberg (2000), S.38f; Tanenbaum (2003a), S.677f

O

OMG *Object Management Group*
Die OMG stellt ein internationales Konsortium dar, das sich mit der Entwicklung von Standards für die objektorientierte Programmierung beschäftigt und dabei vor allem auf Herstellerunabhängigkeit und Systemunabhängigkeit achtet. Zu den bekanntesten Entwicklungen gehören CORBA und UML.
Literatur: Boger (1999), S.87; Sommerville (2007), S.279
Homepage: `http://www.omg.org/` (Letzter Zugriff: 25.07.2007)

OMT *Object Modeling Technique*
(dt. Objektmodellierungstechnik)
Eine objektorientierte Modellierungsnotation von James Rumbaugh, deren Elemente in den heutigen Standard zur Beschreibung objektorientierter Modelle, UML, eingeflossen sind. Literatur: Rumbaugh et al. (1991); Rumbaugh (1996)

P

POP3 *Post Office Protocol Version 3*
Bei POP3 handelt es sich um ein Protokoll zum Herunterladen elektronischer Nachrichten (E-Mails) von einem Server.
Literatur: Kurose & Ross (2005), S.119ff; Tanenbaum (2003a), S.658; Fuhrberg (2000), S.40f

S

SAX *Simple API for XML*
(dt.: Einfache Programmierschnittstelle für XML)
Hierbei handelt es sich um den de facto Standard eines APIs zum Parsen von XML-Dateien. Ursprünglich war SAX nur als Java-API in Form von einer Vielzahl von Interface-Klassen definiert, inzwischen existieren jedoch Implementierungen für nahezu alle gängigen Programmiersprachen.
Literatur: McLaughlin (2001), S.13f; Megginson (2007)
Homepage: `http://www.saxproject.org/` (Letzter Zugriff: 25.07.2007)

SIP *Session Initiation Protocol*
(dt.: Sitzungsaufbauprotokoll)
SIP ist ein Netzprotokoll zum Aufbau einer Kommunikationssitzung zwischen zwei und mehr Teilnehmern, das hauptsächlich in der Internet-Telefonie zum Einsatz kommt.
Literatur: Kurose & Ross (2005), S.602ff; Tanenbaum (2003a), S.745ff

SMTP *Simple Mail Transfer Protocol*
SMTP ist ein Protokoll zur Übertragung von elektronischen Nachrichten (E-Mails) über ein Netzwerk.
Literatur: Fuhrberg (2000), S.41ff; Tanenbaum (2003a), S.655ff; Kurose & Ross (2005), S.112ff

SQL *Structured Query Language*
(dt.: Strukturierte Abragesprache)
Bei SQL handelt es sich um eine standardisierte Programmiersprache, mit der (relationale) Datenbanken erstellt, abgefragt und manipuliert werden können.
Literatur: Elmasri & Navathe (2005), S.181ff; Hamilton et al. (1998), S.25ff

SSH *Secure Shell*
SSH ist ein Programm, mit dessen Hilfe verschlüsselt mit einem Rechner kommuniziert werden kann.
Literatur: Fuhrberg (2000), S.130ff;

SSL *Secure Socket Layer*
SSL ist ein Protokoll, mit dessen Hilfe sichere Verbindungen errichtet und verwendet werden können. Dabei kommen kryptographische Algorithmen für die Ver- und Entschlüsselung zum Einsatz. Obwohl SSL ursprünglich für sichere HTTP-Verbindungen zwischen Web-Browser und -Server konzipiert war (HTTPS), können auch andere Dienste wie SMTP oder POP3 mit SSL abgewickelt werden.
Literatur: Kurose & Ross (2005), S.708ff; Tanenbaum (2003a), S.876ff; Fuhrberg (2000), S.106ff;

SYMLOG *SYstem for the Multiple Level Obervation of Groups*
(dt.: System zur Mehr-Ebenen-Betrachtung von Gruppen)
SYMLOG stellt ein Verfahren für die mehrdimensionale Beobachtung von (Klein-)Gruppen, das von Robert Freed Bales auf der Grundlage von IPA entwickelt wurde.
Literatur: (Bales & Cohen, 1982); Heinze & Farwer (2005)
Homepage: `http://www.symlog.com/` (Letzter Zugriff: 25.07.2007)

U

UML *Unified Modelling Language*
(dt. Vereinheitlichte Modellierungssprache)
Die UML ist eine objektorientierte Modellierungsnotation, die von der OMG entwickelt wird und den heutigen Standard zur Beschreibung objektorientierter Modelle bildet.
Literatur: Fowler & Scott (1998); Zuser et al. (2004), S.193ff
Homepage: `http://www.uml.org/` (Letzter Zugriff: 25.07.2007)

W

W3C *World Wide Web Consortium*
(dt.: WWW-Konsortium)
Das W3C ist ein Gremium, das sich mit der Standardisierung von Techniken befasst, die das Internet betreffen. Beispiele dafür sind HTML und XML.
Homepage: `http://www.w3.org/` (Letzter Zugriff: 25.07.2007)

WBT *Web Based Training*
(dt.: Internet-basiertes Training)
Dieser Begriff umfasst sämtliche Lernformen, die auf der Verwendung von Computern und dem Einsatz des Internet basieren.
Literatur: Kerres (2001), S.13f; Müller & Dürr (2002), S.165ff ; Dittler (2003b), S.153ff

WWW *World Wide Web*
(dt.: Weltweites Netzwerk)
Das WWW ist ein über das Internet abrufbare Hypertext-System, das aus einer Vielzahl miteinander verknüpften Web-Seiten (in HTML) besteht.
Synonyme: Web, WWW oder auch Internet
Literatur: Fuhrberg (2000), S.241ff; Tanenbaum (2003a), S.664ff; Döring (2003), S.8ff; Teufel et al. (1995)), S.165ff

X

XML *Extensible Markup Language*
(dt.: erweiterbare Auszeichnungssprache)
Die XML ist eine Beschreibungssprache, mit welcher sich in einem Textformat hierarchisch strukturierte Daten notieren lassen. Ein bevorzugtes Einsatzgebiet von XML liegt im Austausch von Daten zwischen unterschiedlichen Computer-Systemen über das Internet.

Literatur: McLaughlin (2001); Mintert (2007)
Homepage: `http://www.w3.org/XML/` (Letzter Zugriff: 25.07.2007)

Literaturverzeichnis

[Aho et al. 1988a] Aho, Alfred V. ; Sethi, Ravi ; Ulman, Jeffrey D.: *Compilerbau*. Bd. 1. München : Addison-Wesley, 1988

[Aho et al. 1988b] Aho, Alfred V. ; Sethi, Ravi ; Ulman, Jeffrey D.: *Compilerbau*. Bd. 2. München : Addison-Wesley, 1988

[Aliev et al. 2000] Aliev, Rafik ; Bonfig, Karl W. ; Aliew, Fuad: *Soft-Computing – Eine grundlegende Einführung*. Berlin : Verlag Technik, 2000

[Allen 2002] Allen, Eric: *Bug Patterns In Java*. Berkeley (CA) : Apress, 2002. – ISBN 1-59059-061-9

[Appelt 2004] Appelt, Wolfgang: *Plattformen*. Kap. 2.2, S. 137–153. Siehe (Haake et al., 2004)

[Austin 2002] Austin, John L.: *Zur Theorie der Sprechakte (How to do things with Words)*. Stuttgart : Reclam, 2002

[Axelrod 1976] Axelrod, Robert: *Structure of decision. The cognitive maps of political elites*. Princeton, NJ : Princeton University Press, 1976

[Badach et al. 2003] Badach, Anatol ; Rieger, Sebastian ; Schmauch, Matthias: *Web-Technologien – Architekturen, Konzepte, Trends*. München : Hanser, 2003

[Baker & Lund 1996] Baker, Michael J. ; Lund, Kristine: Flexibly structuring the interaction in a CSCL environment. In: Brna, P. (Hrsg.) ; Paiva, A. (Hrsg.) ; Self, J. (Hrsg.): *Proceedings of the EuroAIED Conference*. Lisbon : Edicoes Colibri, 1996, S. 401–407. – URL `http://www.vjf.cnrs.fr/umr8606/FichExt/mbaker/publications/ArticlesBakerPDF/1996/1996EtAl-b.pdf`. – Zugriffsdatum: 21.06.2005

[Baker et al. 2001] Baker, Michael ; Vries, E. de ; Lund, Kristine ; Quignard, M.: Computer-mediated epistemic interactions for co-constructing scientific notions. In: Dillenbourg, P. (Hrsg.) ; Eurelings, A. (Hrsg.) ; Hakkarainen, K. (Hrsg.): *European perspectives on computer-supported collaborative learning : proceedings of the first European conference on computer-supported collaborative learning*, Unigraphic, 2001, S. 89–96. – URL `http://web.upmf-grenoble.fr/sciedu/edevries/Baker\_etal\_EuroCSCL.pdf`. – Zugriffsdatum: 16.06.2005

[Baker & Lund 1997] Baker, M.J. ; Lund, K.: Promoting reflective interactions in a computer-supported collaborative learning environment. In: *Journal of Computer Assisted Learning* (1997), Nr. 13, S. 175–193. – URL `http://sir.univ-lyon2.fr/GRIC/GRIC5/Home/mbaker/webpublications/Baker-LundJCAL.PDF`. – Zugriffsdatum: 31.07.2003

[Bales & Cohen 1982] Bales, Robert F. ; Cohen, Stephen P.: *SYMLOG – Ein System für die mehrstufige Beobachtung von Gruppen*. Stuttgart : Klett-Cotta, 1982

[Bales 1950] Bales, Robert F.: *Interaction Process Analysis – A Model for the Study of Small Groups*. Chicago – London : The University of Chicago Press, 1950

[Balzert 1999] Balzert, Helmut: *Lehrbuch Grundlagen der Informatik*. Heidelberg : Spektrum, 1999

[Baumgartner & Payr 1994] Baumgartner, Peter ; Payr, Sabine: *Lernen mit Software*. Innsbruck : Studien Verlag, 1994

[Bendel 2003] Bendel, Oliver: *Pädagogische Agenten im Corporate E-Learning*, Universität St. Gallen, Dissertation, 2003

[Bischoff 2005] Bischoff, Kerstin: *Objektorientierte Softwareentwicklung in virtuellen Teams – Modellierung und Ansätze zur automatischen Erkennung von Problemsituationen*, Universität Hildesheim, Diplomarbeit, 2005

[Bishop 2000] Bishop, Judy: *Java lernen – Anfangen, Anwenden, Verstehen*. München : Addison-Wesley, 2000. – ISBN 3-8273-1605-7

[Boger 1999] Boger, Marko: *Java in verteilten Systemen – Nebenläufigkeit, Verteilung, Persistenz*. Heidelberg : dpunkt.verlag GmbH, 1999. – ISBN 3-932588-32-0

[du Boulay 1989] Boulay, Benedict du: Some Difficulties in Learning to Program. In: *Journal of Educational Computing Research* (1989), Nr. 2, S. 57–73

[Breuer 2001] Breuer, Jens: *Kooperative Lernformen beim E-Learning einsetzen*. Kap. 4.2. Siehe (Hohenstein & Wilbers, 2005)

[Broy & Spaniol 1999] Broy, Manfred (Hrsg.) ; Spaniol, Otto (Hrsg.): *VDI-Lexikon Informatik und Kommunikationstechnik*. 2., erweiterte und neubearbeitete Auflage. Berlin : Springer, 1999

[Büning & Trenkler 1979] Büning, Herbert ; Trenkler, Götz: *Nichtparametrische statistische Methoden*. Berlin : de Gruyter, 1979. – ISBN 3-110-08134-2

[Campione & Walrath 1998a] Campione, Mary ; Walrath, Kathy: *The Java Tutorial – Object-Oriented Programming for the Internet*. 2nd edition. Reading, Massachusetts : Addison-Wesley, 1998. – ISBN 0-201-31007-4

[Campione & Walrath 1998b] Campione, Mary ; Walrath, Kathy: *The JFC Swing Tutorial – A Guide to Constructin GUIs*. Reading, Massachusetts : Addison-Wesley, 1998. – ISBN 0-201-43321-4

[Clauss et al. 1995] Clauss, Günter ; Finze, Falk-Rüdiger ; Partzsch, Lothar: *Statistik für Soziologen, Pädagogen, Psychologen und Mediziner – Grundlagen*. Bd. 1. 2., überarbeitet und erweiterte Auflage. Frankfurt am Main : Verlag Harri Deutsch, 1995. – ISBN 3-8171-1415-X

[Comer 2002] Comer, Douglas E.: *Computernetzwerke und Internets mit Internet-Anwendungen*. 3., überarbeitete Auflage. München : Prentice Hall, 2002. – ISBN 3-8273-7023-X

[Constantino-Gonz'alez & Suthers 2001] Constantino-Gonz'alez, María de los A. ; Suthers, Daniel D.: *Coaching Collaboration by Comparing Solutions and Tracking Participation*. 2001. – URL `http://lilt.ics.hawaii.edu/lilt/papers/2001/Constantino-Suthers-Euro-CSCL-2001.pdf`. – Zugriffsdatum: 19.08.2003

[Davies et al. 2001] Davies, Jim R. ; Gertner, Abigail S. ; Lesh, Neal ; Rich, Charles ; Sidner, Candace L. ; Rickel, Jeff: Incorporating tutorial strategies into an intelligent assistant. In: *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*. New York, NY, USA : ACM Press, 2001, S. 53–56. – ISBN 1-58113-325-1

[Dawabi 2004] Dawabi, Peter: *Virtuelle kooperative Lernräume*. Kap. 2.1.6, S. 118–126. Siehe (Haake et al., 2004)

[Deitel 2005] Deitel, Harvey M. Deitel; Paul J.: *Java – How to program*. 6. Auflage. New Jersey : Pearson Education, 2005

[Dellwig 2000] Dellwig, Ingo: *HTML 4 Nitty Gritty*. München : Addison-Wesley, 2000

[Dittler 2003a] Dittler, Ullrich: *Erfolgsfaktor Multimedia-Didaktik – Drei Beispiele*. Kap. 5, S. 75–91. Siehe (Dittler, 2003b)

[Dittler 2003b] Dittler, Ullrich (Hrsg.): *E-Learning – Einsatzkonzepte und Erfolgsfaktoren des Lernens mit interaktiven Medien*. München – Wien : Oldenbourg, 2003

[Domeshek et al. 2004] Domeshek, Eric A. ; Holman, Elias ; Luperfoy, Susann: *Discussion Control in an Automated Socratic Tutor*. 2004. – URL `http://www.stottlerhenke.com/papers/IITSEC-04-socratic-discussion.pdf`. – Zugriffsdatum: 20.09.2005

[Döring 2003] Döring, Nicola: *Sozialpsychologie des Internet – Die Bedeutung des Internet für Kommunikationsprozesse, Identitäten, soziale Beziehungen und Gruppen*. 2. vollständig überarbeitete und erweiterte Auflage. Göttingen – Bern – Toronto – Seattle : Hogrefe – Verlag für Psychologie, 2003

[Dorn & Gottlob 1999] Dorn, Jürgen ; Gottlob, Georg: *Künstliche Intelligenz*. Kap. E7, S. 975–998. Siehe (Rechenberg & Pomberger, 1999)

[Dorsch 1982] Dorsch, Friedrich (Hrsg.): *Psychologisches Wörterbuch*. 10., neubearbeitete Auflage. Bern – Stuttgart – Wien : Hans Huber, 1982

[Effelsberg et al. 2004] Effelsberg, Wolfgang ; Liebig, Hans C. ; Scheele, Nicolai ; Vogel, Jürgen: *Kooperationen in größeren Lerngruppen*. Kap. 2.1.4, S. 96–108. Siehe (Haake et al., 2004)

[Elmasri & Navathe 2005] Elmasri, Ramez ; Navathe, Shamkant B.: *Grundlagen von Datenbanksystemen*. 3. Auflage. München : Addison-Wesley, 2005. – ISBN 3-8273-7153-8

[Engelbart 1962] Engelbart, Douglas C.: Augmenting Human Intellect: A Conceptual Framework. (1962). – URL `http://www.bootstrap.org/augdocs/friedewald030402/augmentinghumanintellect/ahi62index.html`. – Zugriffsdatum: 25.07.2007

[Eunson 1990] Eunson, Baden: *Betriebspsychologie*. Hamburg – New York : MacGraw-Hill, 1990

[Feltz et al. 2006] Feltz, Fernand (Hrsg.) ; Otjacques, Benoît (Hrsg.) ; Oberweis, Andreas (Hrsg.) ; Poussing, Nicolas (Hrsg.): *AIM 2006 – Information Systems and Collaboration: State of the Art and Perspectives, Best Papers of the 11th International Conference of the Association Information and Management (AIM), Luxembourg, June 8-9, 2006*. Bd. *92*. GI, 2006. (LNI). – ISBN 978-3-88579-186-7

[Ferber 2001] Ferber, Jacques: *Multiagentensysteme*. München : Addison-Wesley, 2001

[Fisch 1994] Fisch, Rudolf: Eine Methode zur Analyse von Interaktionsprozessen beim Problemlösen in Gruppen. In: *Gruppendynamik* (1994), Nr. 25, S. 149–168

[Floyd & Züllighoven 1999] Floyd, Christiane ; Züllighoven, Heinz: *Softwaretechnik*. Kap. D15, S. 763–790. Siehe (Rechenberg & Pomberger, 1999)

[Fowler & Scott 1998] Fowler, Martin ; Scott, Kendall: *UML konzentriert – Die neue Standard-Objektmodellierungssprache anwenden*. München : Addison-Wesley, 1998. – ISBN 3-8273-1329-5

[Fuhrberg 2000] Fuhrberg, Kai: *Internet-Sicherheit – Browser, Firewalls und Verschlüsselung*. 2., aktualisierte Auflage. München-Wien : Hanser, 2000

[Gaensslen & Schubö 1976] Gaensslen, Hermann ; Schubö, Werner: *Einfache und komplexe statistische Analyse*. 2. verbesserte Auflage. München : Ernst Reinhardt Verlag, 1976. – ISBN 3-497-00705-6

[Gamma et al. 1996] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software*. München : Addison-Wesley, 1996

[Gerbner 1989] Gerbner, George: *A generalized graphic model of communication*. Kap. 2, S. 17–18. In: Corner, John (Hrsg.) ; Hawthorn, Jeremy (Hrsg.): *Communication Studies – An Introductory Reader*. London : Edward Arnold, 1989

[Göldner 2005] Göldner, Antje: *Computerunterstützte Kommunikation in virtuellen Teams – Klassifikations- und Lösungsansätze für Problemsituationen in der Chatkommunikation um Rahmen objektorientierter Programmierung*, Universität Hildesheim, Diplomarbeit, 2005

[Goos & Zimmermann 1999] Goos, Gerhard ; Zimmermann, Wolf: *Programmiersprachen*. Kap. D2, S. 469–515. Siehe (Rechenberg & Pomberger, 1999)

[Gross & Koch 2007] Gross, Tom ; Koch, Michael: *Computer-Supported Cooperative Work*. München : Oldenbourg, 2007. – ISBN 978-3-486-58000-6

[Grune & de Witt 2004] Grune, Christian ; Witt, Claudia de: *Pädagogische und didaktische Grundlagen*. Kap. 1.4, S. 27–41. Siehe (Haake et al., 2004)

[Haake et al. 2004] Haake, Jörg M. (Hrsg.) ; Schwabe, Gerhard (Hrsg.) ; Wessner, Martin (Hrsg.): *CSCL-Kompendium – Lehr- und Handbuch zum computerunterstützten kooperativen Lernen*. München : Oldenbourg, 2004

[Hagenhoff et al. 2001] Hagenhoff, Svenja ; Schumann, Matthias ; Schellhase, Jörg: *Lernplattformen auswählen*. Kap. 45.1. Siehe (Hohenstein & Wilbers, 2005)

[von Hagen 2005] Hagen, Ulrich von: Das neue V-Modell XT. In: *LDVZ-Nachrichten* (2005), Nr. 1, S. 29–37. – URL `ftp://ftp.uni-kl.de/pub/v-modell-xt/Release-1.1/Infomaterial/Das-neue-V-Modell.pdf`. – Zugriffsdatum: 25.11.2005

[Hamilton et al. 1998] Hamilton, Graham ; Cattell, Rick ; Fisher, Maydene: *JDBC – Datenbankzugriff mit Java*. München : Addison-Wesley, 1998. – ISBN 3-8273-1306-6

[Hansen 2003] Hansen, Sabine: *Interaktion als Erfolgsfaktor virtueller Seminare - Erfahrungen der niederländischen Network University*. Kap. 4.3.3. Siehe (Hohenstein & Wilbers, 2005)

[Hare et al. 1955] Hare, Paul A. (Hrsg.) ; Birgatta, Edgar F. (Hrsg.) ; Bales, Robert F. (Hrsg.): *Small Groups – Studies in Social Interaction*. New York : Alfred A. Knopf, 1955

[Hare 1962] Hare, Paul A.: *Handbook of Small Group Research*. New York : The Free Press of Glencoe, 1962

[Hasenkamp et al. 1994] Hasenkamp, Ulrich (Hrsg.) ; Kirn, Stefan (Hrsg.) ; Syring, Michael (Hrsg.): *CSCW – Computer Supported Cooperative Work*. Bonn : Addison-Wesley, 1994

[Hégaret et al. 2005] Hégaret, Philippe L. ; Whitmer, Ray ; Wood, Lauren: *W3C Document Object Model*. 2005. – URL `http://www.w3.org/DOM/`. – Zugriffsdatum: 24.07.2007

[Heinze & Farwer 2005] Heinze, Frank ; Farwer, Heiko: *Quantitative Methoden der Organisationsforschung – Beobachtung mit SYMLOG*. 2005. – URL `http://www.qualitative-research.net/organizations/2/or-bs-d.htm`. – Zugriffsdatum: 25.07.2007

[Herget et al. 1999] Herget, Josef ; Hörmann, Stefan ; Lang, Norbert ; Kuhlen, Rainer: *Analyse der Nutzeranforderungen an Electronic Publishing Produkte. State-of-the-Art-Report*. In: *Die Buchbranche in der Informationsgesellschaft: Nutzeranforderungen an elektronische Medien in Verlagen, Buchhandlungen und Bibliotheken*. Berlin : International Book Agency, 1999

[Hoffmann 2004] Hoffmann, Nicole: *Problemorientiertes Lernen*. Kap. 3.4.3, S. 245–251. Siehe (Haake et al., 2004)

[Hohenstein & Sander 2005] Hohenstein, Andreas ; Sander, Elisabeth: *Konfliktinduzierenden Lernen mit Standard-Lernprogrammen*. Kap. 4.23. Siehe (Hohenstein & Wilbers, 2005)

[Hohenstein & Wilbers 2005] Hohenstein, Andreas (Hrsg.) ; Wilbers, Karl (Hrsg.): *Handbuch E-Learning: Expertenwissen aus Wissenschaft und Praxis*. Köln : Dt. Wirtschaftsdienst, 2005

[Holmer et al. 2001] Holmer, Torsten ; Haake, Jörg ; Streitz, Norbert: *Kollaborationsorientierte synchrone Werkzeuge*. Kap. 2.1, S. 180–193. Siehe (Schwabe et al., 2001). – URL `http://www.ifi.unizh.ch/im/imrg/fileadmin/publications/Koordinationswerkzeuge.pdf`. – Zugriffsdatum: 03.01.2006

[Holmer & Jödick 2004] Holmer, Torsten ; Jödick, Friederike: *Kooperation in kleineren Lerngruppen*. Kap. 2.1.3, S. 86–95. Siehe (Haake et al., 2004)

[Hopcroft et al. 2002] Hopcroft, John E. ; Motwani, Rajeev ; Ullman, Jeffrey D.: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. 2., überarbeitete Auflage. München : Addison-Wesley, 2002

[Hupfeld-Heinemann 2001] Hupfeld-Heinemann, Jörg: *Die systematische Analyse sozialer Interaktionen*. 2001. – URL `http://www.psy.unibe.ch/soz/team/pdf/hupfeld/Hupfeld2001.pdf`. – Zugriffsdatum: 15.08.2005

[Jermann 1999] Jermann, Patrick: *Structuring and regulating collaborative interaction by semi-structured interfaces and interaction meters*. 1999. – URL `http://cbl.leeds.ac.uk/~tamsin/dialogueworkshop/jermann-interfaces.pdf`. – Zugriffsdatum: 31.07.2003

[Johnson & Johnson 1991] Johnson, David W. ; Johnson, Roger T.: *Learning together and alone*. Englewood Cliffs, NJ : Prentice Hall, 1991

[Johnson & Murch 2000] Johnson, Tony ; Murch, Richard: *Agententechnologie: Die Einführung*. München : Addison-Wesley, 2000

[Kastens & Kleine Büning 2005] Kastens, Uwe ; Kleine Büning, Hans: *Modellierung – Grundlagen und formale Methoden*. München : Carl Hanser Verlag, 2005. – ISBN 3-446-40460-0

[Kauffeld 2001] Kauffeld, Simone: *Teamdiagnose*. Göttingen – Bern – Toronto – Seattle : Hogrefe – Verlag für Psychologie, 2001

[KBSt 2004] KBSt ; Innern (KBSt), Koordinierungs und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des (Hrsg.): *V-Modell-Referenz Rollen*. 2004. – URL `http://www.kbst.bund.de/V-Modell/V-Modell-XT-,395/Dokumentation.htm#Rollen`. – Zugriffsdatum: 08.11.2005

[Kerres 2001] Kerres, Michael ; Kerres, Michael (Hrsg.): *Multimediale und telemediale Lernumgebungen*. Bochum : Oldenbourg, 2001

[Klauser 2002] Klauser, Fritz: *E-Learning problembasiert gestalten*. Kap. 4.12. Siehe (Hohenstein & Wilbers, 2005)

[Klauss 2005] Klauss, Sonja-Maria: *Didactics, Architecture and Design of e-Learning Environments – A Preparatory Contribution to the International e-Creator Project funded by the European Commission*, Universität Hildesheim, Diplomarbeit, 2005

[Klügl 2001] Klügl, Franziska: *Multiagentensimulation*. München : Addison-Wesley, 2001

[Koch & Gross 2006] Koch, Michael ; Gross, Tom: Computer-supported cooperative work – concepts and trends. In: (Feltz et al., 2006), S. 165–172. – URL `http://www.communixx.de/files/Koch2006-aim.pdf`. – Zugriffsdatum: 28.11.2007. – ISBN 978-3-88579-186-7

[Koch 2001] Koch, Michael: *Community-Support-Systeme*. Kap. 2.3, S. 286–296. Siehe (Schwabe et al., 2001). – URL `http://www.ifi.unizh.ch/im/imrg/fileadmin/publications/Koordinationswerkzeuge.pdf`. – Zugriffsdatum: 03.01.2006

[Kölle & Langemeier 2005] Kölle, Ralph ; Langemeier, Glenn: *Rollen in virtuellen Teams – Analyse und Simulation (Arbeitsbericht vom 21.12.2005).* 2005. – URL `http://www.vitaminl.de/downloads/arbeitsbericht2005-12-21.pdf`. – Zugriffsdatum: 08.08.2007

[Kölle 2007] Kölle, Ralph: *Virtuelle Mitglieder in virtuellen Teams – Kompensation defizitärer Rollen durch Simulation.* Boizenburg, Universität Hildesheim, Dissertation, 2007

[Konradt & Hertel 2002] Konradt, Udo ; Hertel, Guido: *Management virtueller Teams – Von der Telearbeit zum virtuellen Unternehmen.* Weinheim und Basel : Beltz, 2002

[Konrad & Traub 2005] Konrad, Klaus ; Traub, Silke: *Kooperatives Lernen – Theorie und Praxis in Schule, Hochschule und Erwachsenenbildung.* 2., überarbeitet und ergänzte Auflage. Baltmannsweiler : Schneider Verlag Hohengehren GmbH, 2005

[Kriz & Nöbauer 2002] Kriz, Willy C. ; Nöbauer, Brigitta: *Teamkompetenz – Konzepte, Trainingsmethoden, Praxis.* Göttingen : Vandenhoeck & Ruprecht, 2002

[Krüger 2006] Krüger, Guido: *Handbuch der Java-Programmierung.* 4. Auflage. München : Addison-Wesley, 2006. – URL `http://www.javabuch.de/`. – Zugriffsdatum: 24.05.2007. – ISBN 978-3-8273-2447-4

[Kuhlen et al. 2004] Kuhlen, Rainer (Hrsg.) ; Seeger, Thomas (Hrsg.) ; Strauch, Dietmar (Hrsg.): *Grundlagen der praktischen Information und Dokumentation.* Bd. 2. 5., völlig neu gefasste Ausgabe. München : Saur, 2004

[Kühnel 2001] Kühnel, Ralf: *Agentenbasierte Softwareentwicklung.* München : Addison-Wesley, 2001

[Kurose & Ross 2002] Kurose, James F. ; Ross, Keith W.: *Computernetze – Ein Top-Down-Ansatz mit Schwerpunkt Internet.* München : Addison-Wesley, 2002. – ISBN 3-8273-7017-5

[Kurose & Ross 2005] Kurose, James F. ; Ross, Keith W.: *Computer networking – A top-down approach featuring the internet.* 3rd edition. Boston : Addison-Wesley, 2005. – ISBN 0-321-26976-4

[Lang 1978] Lang, Norbert: *Lehrer und Fernsehen. Überlegungen und Untersuchungen zur Rolle der öffentlichen Erziehung im Prozess der Massenkommunikation. Dargestellt am Beispiel Fernsehen.* München : Minerva, 1978

[Lang 2002] Lang, Norbert: *Lernen in der Informationsgesellschaft – Mediengestütztes Lernen im Zentrum einer neuen Lernkultur.* S. 23–42. Siehe (Scheffer & Hesse, 2002)

[Leavitt 1951] Leavitt, H.J.: Some effects of certain communication patterns on group performance. In: *Journal of Abnormal and Social Psychology* (1951), Nr. 46, S. 38–50

[Liang 2005] Liang, Y. D.: *Introduction to Java Programming – Comprehensive Version*. 5th edition. New Jersey : Prentice Hall, 2005

[Lindgren 1973] Lindgren, Henry C.: *Einführung in die Sozialpsychologie*. Weinheim : Beltz, 1973

[Lopez-Herrejon & Schulman 2004] Lopez-Herrejon, Roberto E. ; Schulman, Morrie: Using interactive technology in a short java course: an experience report. In: *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*. New York, NY, USA : ACM Press, 2004, S. 203–207. – ISBN 1-58113-836-9

[Lund 1999] Lund, Michael J.: Teachers' collaborative interpretations of students' computer-mediated collaborative problem solving interactions. In: Vivet, S.P. (Hrsg.): *Artificial Intelligence in Education*, IOS Press, 1999, S. 147–154. – URL `http://eprints.ens-lsh.fr/archive/00000130/01/Lund-BakerAIED99.pdf`. – Zugriffsdatum: 16.06.2005

[Macha 2001] Macha, Hildegard: *Lernstile diagnostizieren und individuelle Potenziale fördern*. Kap. 4.4. Siehe (Hohenstein & Wilbers, 2005)

[Mandl 2004] Mandl, Heinz: E-Learning – Trends und zukünftige Entwicklungen. In: Rebensburg, Klaus (Hrsg.): *Grundfragen Multimedialen Lehrens und Lernens*, 2004, S. 17–29. – ISBN 3-8334-1573-8

[Mann 2001] Mann, Leon: *Sozialpsychologie*. Weinheim : Beltz, 2001

[Margerison et al. 1995] Margerison, Charles ; McCann, Dick ; Davies, Rod: Focus on team appraisal. In: *Team Performance Management* 1 (1995), Nr. 4, S. 13–18. – URL `http://www.apmforum.com/tpi.pdf`. – Zugriffsdatum: 17.11.2005

[Margerison 2005] Margerison, Charles: How do you prefer to work? In: *Training Journal* (2005), S. 22–26. – URL `http://www.viprojects.com/userfiles/documents/HowDoYouPrefertoWork-Margerison-TrainingJournalJune2005.pdf`. – Zugriffsdatum: 17.11.2005

[Martens 2003] Martens, Jens U.: *Der Persönliche Berater – Förderung erfolgsbestimmter Einstellungen*. Kap. 7, S. 121–138. Siehe (Dittler, 2003b)

[Matessa 2001] Matessa, Michael: *The Benefit of Structured Interfaces in Collaborative Communication*. 2001. – URL `http://www.matessa.org/~mike/pubs/aaai01.pdf`. – Zugriffsdatum: 31.07.2003

[McLaughlin 2001] McLaughlin, Brett: *Java und XML*. Köln : O'Reilley, 2001

[McManus & Aiken 1995] McManus, Margaret M. ; Aiken, Robert M.: Monitoring computer-based problem solving. In: *Journal of Artificial Intelligence in Education* (1995), Nr. 6, S. 307–336

[McManus 1997] McManus, Margaret M.: Computer supported collaborative learning. In: *SIGGROUP Bull.* 18 (1997), Nr. 1, S. 7–9. – URL `http://portal.acm.org/ft\_gateway.cfm?id=271161\&type=pdf\&coll=GUIDE\&dl=GUIDE\&CFID=54538156\&CFTOKEN=73567581`. – Zugriffsdatum: 13.09.2005

[Megginson 2007] Megginson, David: *SAX*. 2007. – URL `http://www.saxproject.org/`. – Zugriffsdatum: 24.07.2007

[Mintert 2007] Mintert, Stefan: *edition W3C.de – Extensible Markup Language (XML) 1.1*. 2007. – URL `http://www.edition-w3c.de/TR/2004/REC-xml11-20040204/`. – Zugriffsdatum: 24.07.2007

[Müller & Dürr 2002] Müller, Roman ; Dürr, Johannes: *Plattformen und Programme – Grundlegende Verfahren und Tools des E-Learning*. S. 164–184. Siehe (Scheffer & Hesse, 2002)

[Park & Lee 2003] Park, Ok-choon ; Lee, Jung: *Adaptive Instructional Systems*. 2003

[Park & Bang 2002] Park, Won-Woo ; Bang, Hojin: Team Role Balance and Team Performance. In: *Belbin Biennal Conference* (2002). – URL `http://www.belbin.com/Conf2002/Belbin\%202002-Team\%20role\%20balance\%20and\%20team\%20effectiveness\%20(paper).pdf`. – Zugriffsdatum: 17.11.2005

[Petschenka et al. 2004] Petschenka, Anke ; Ojstersek, Nadine ; Kerres, Michael: *Lernaufgaben gestalten - Lerner aktivieren mit didaktisch sinnvollen Lernaufgaben*. Kap. 4.19. Siehe (Hohenstein & Wilbers, 2005)

[Pfister & Wessner 2001a] Pfister, Hans-Rüdiger ; Wessner, Martin: Communities of Learners: Vom kooperativen Lernen zum kooperativen Wissensmanagement. In: *Limpact: Leitprojekte Informationen Compact* (2001), Nr. 2, S. 7–12. – URL `http://www.bibb.de/de/limpact13175.htm`. – Zugriffsdatum: 24.08.2005. – ISSN 1439-8079

[Pfister & Wessner 2001b] Pfister, Hans-Rüdiger ; Wessner, Martin: *Kooperatives Lehren und Lernen*. S. 251–263. Siehe (Schwabe et al., 2001). – URL `ftp://ftp.ipsi.fraunhofer.de/pub/concert/publications/kapitel-cscl-final.pdf`. – Zugriffsdatum: 24.08.2005

[Pfister 2000] Pfister, Hans-Rüdiger: Kooperatives compunterunterstütztes Lernen (CSCL) – Was ist das und wozu nützt es? Eröffnung des CSCL-Kompetenzzentrums am GMD-IPSI in Darmstadt. In: *nfd Information – Wissenschaft und Praxis* (2000), Nr. 4 (51. Jg.), S. 227–231

[Plattner & Schulthess 1999] Plattner, Bernhard ; Schulthess, Peter: *Rechnernetze*. Kap. A3, S. 381–407. Siehe (Rechenberg & Pomberger, 1999)

[Puppe 1992] Puppe, Frank: Intelligente Tutorsysteme. In: *Informatik Spektrum* 15 (1992), Nr. 4, S. 195–207

[Rechenberg & Pomberger 1999] Rechenberg, Peter (Hrsg.) ; Pomberger, Gustav (Hrsg.): *Informatik-Handbuch*. 2., aktualisierte und erweiterte Auflage. München : Hanser, 1999

[Rechenberg 1999] Rechenberg, Peter: *Formale Sprachen und Automaten*. Kap. A3, S. 89–110. Siehe (Rechenberg & Pomberger, 1999)

[Reichling et al. 2004] Reichling, Tim ; Becks, Andreas ; Bresser, Oliver ; Wulf, Volker: *Koordinationswerkzeuge zur Bildung von Lerngruppen*. Kap. 2.1.2, S. 80–85. Siehe (Haake et al., 2004)

[Rembold & Levi 1999] Rembold, Ulrich ; Levi, Paul: *Einführung in die Informatik für Naturwissenschaftler und Ingenieure*. 3., vollständig überarbeitet und erweiterte Auflage. München : Carl Hanser Verlag, 1999. – ISBN 3-446-18157-1

[Rickel et al. 2002] Rickel, Jeff ; Gertner, Abigail ; Lesh, Neal ; Rich, Charles ; Sidner, Candace L.: *Collaborative Discourse Theory as a Foundation for Tutorial Dialogue*. Juni 2002. – URL `http://www.isi.edu/isd/rickel/its02.pdf`. – Zugriffsdatum: 22.08.2002

[Röhr et al. 1983] Röhr, Michael ; Lohse, Heinz ; Ludwig, Rolf: *Statistik für Soziologen, Pädagogen, Psychologen und Mediziner – Statistische Verfahren*. Bd. 2. Frankfurt am Main : Verlag Harri Deutsch, 1983. – ISBN 3-87144-596-7

[Rosemann & Bielski 2001] Rosemann, Bernhard ; Bielski, Sven: *Pädagogische Psychologie*. Weinheim und Basel : Beltz, 2001

[Rüger 2002] Rüger, Bernhard: *Test- und Schätztheorie – Statistische Tests*. Bd. 2. München : Oldenbourg, 2002. – ISBN 3-486-25130-9

[Rumbaugh et al. 1991] Rumbaugh, James ; Blaha, Michael ; Premerlani, William ; Eddy, Frederick ; Lorensen, William ; Rumbaugh, James (Hrsg.): *Object-oriented modeling and design*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1991. – ISBN 0-13-629841-9

[Rumbaugh 1996] Rumbaugh, James: *OMT Insights*. New York : SIGS Books, 1996

[Russell & Norvig 2004] Russell, Stuart ; Norvig, Peter: *Künstliche Intelligenz – Ein moderner Ansatz*. 2. Auflage. München : Pearson Studium, 2004

[Sader 2002] Sader, Manfred: *Psychologie der Gruppe*. 8. Auflage. Weinheim : Juventa, 2002

[Savitch 2007] Savitch, Walter: *Absolute Java*. 3rd edition. New Jersey : Pearson Education, 2007

[Schaumburg 2003] Schaumburg, Heike: *Konstruktivistischer Unterricht mit Laptops? Eine Fallstudie zum Einfluss mobiler Computer auf die Methodik des Unterrichts*, FU Berlin, Dissertation, 2003. – URL `http://www.diss.fu-berlin.de/2003/63/index.html`. – Zugriffsdatum: 15.09.2005

[Scheffer & Hesse 2002] Scheffer, Ute (Hrsg.) ; Hesse, Friedrich W. (Hrsg.): *CSCL-Kompendium – Lehr- und Handbuch zum computerunterstützten kooperativen Lernen*. Klett-Cotta, 2002

[Schenk 2004] Schenk, Birgit: *Moderation*. Kap. 3.2, S. 208–218. Siehe (Haake et al., 2004)

[Schiedermeier 2005] Schiedermeier, Reinhard: *Programmieren in Java – Eine methodische Einführung*. München : Pearson Studium, 2005

[Schill 2007] Schill, Katrin: *Lassen sich optimale Gruppenzusammensetzungen für synchrone Javaprogrammierung in virtuellen Teams festlegen?* 2007

[Schlienger-Merki & Schauer 2004] Schlienger-Merki, Claudia ; Schauer, Helmut: *Coaching*. Kap. 3.3, S. 219–228. Siehe (Haake et al., 2004)

[Schmid & Caspari 1998] Schmid, Bernd ; Caspari, Sabine: *Zugänge zur Wirklichkeit – Die Typenlehre nach C.G. Jung*. 1998. – URL `http://www.systemische-professionalitaet.de/download/schriften/72-zugaenge-zur-wirklichkeit.pdf`. – Zugriffsdatum: 23.11.2005

[Schubert et al. 2002] Schubert, Sigrid E. (Hrsg.) ; Reusch, Bernd (Hrsg.) ; Jesse, Norbert (Hrsg.): *Informatik bewegt: Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.v. (GI), 30. September – 3.Oktober 2002 in Dortmund*. Bd. *19*. GI, 2002. (LNI). – ISBN 3-88579-348-2

[Schulmeister 1997] Schulmeister, Rolf: *Grundlagen hypermedialer Lernsysteme: Theorie – Didaktik – Design*. München : Oldenbourg, 1997

[Schulmeister 2001] Schulmeister, Rolf: *Virtuelle Universität – Virtuelles Lernen*. München : Oldenbourg, 2001

[Schümmer & Haake 2004] Schümmer, Till ; Haake, Jörg M.: *Kommunikation*. Kap. 2.1.1, S. 66–79. Siehe (Haake et al., 2004)

[Schwabe et al. 2001] Schwabe, Gerhard (Hrsg.) ; Streitz, Norbert (Hrsg.) ; Unland, Rainer (Hrsg.): *CSCW-Kompendium – Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten*. München : Springer, 2001

[Schwabe & Streitz 2001] Schwabe, Gerhard ; Streitz, Norbert ; Unland, Rainer (Hrsg.): *CSCW-Kompendium – Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten*. Berlin : Springer, 2001. – ISBN 3-540-67552-3

[Searle 1983] Searle, John R.: *Sprechakte – Ein sprachphilosophischer Essay*. Frankfurt am Main : Suhrkamp, 1983

[Seibt 2001] Seibt, Dietrich: *Kosten und Nutzen des E-Learning bestimmen*. Kap. 3.3. Siehe (Hohenstein & Wilbers, 2005)

[Seufert & Wessner 2004] Seufert, Sabine ; Wessner, Martin: *Werkzeuge für spezielle Lernmethoden*. Kap. 2.1.7, S. 127–136. Siehe (Haake et al., 2004)

[Singley et al. 2000] Singley, Mark K. ; Singh, Moninder ; Fairweather, Peter ; Farrell, Robert ; Swerling, Steven: Algebra jam: supporting teamwork and managing roles in a collaborative learning environment. In: *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, ACM Press, 2000, S. 145–154. – URL `http://doi.acm.org/10.1145/358916.358985`. – Zugriffsdatum: 10.02.2003. – ISBN 1-58113-222-0

[Soller & Busetta 2003] Soller, Amy ; Busetta, Paolo: *An Intelligent Agent Architecture for Facilitating Knowledge Sharing Communication*. 2003. – URL `http://www.traclabs.com/~cmartin/hmas/wkshp\_2003/papers/Soller.pdf`. – Zugriffsdatum: 19.08.2003

[Soller et al. 1998] Soller, Amy ; Goodman, Bradley A. ; Linton, Frank ; Gaimari, Robert: Promoting Effective Peer Interaction in an Intelligent Collaborative Learning System. In: *ITS '98: Proceedings of the 4th International Conference on Intelligent Tutoring Systems*. London, UK : Springer-Verlag, 1998, S. 186–195. – URL `http://portal.acm.org/citation.cfm?id=648029.745668#`. – Zugriffsdatum: 15.03.2006. – ISBN 3-540-64770-8

[Soller et al. 1999] Soller, Amy ; Lesgold, Alan ; Goodman, Bradley A. ; Linton, Frank: *What Makes Peer Interaction Effective? Modeling Effective Communication in an Intelligent CSCL*. 1999. – URL `http://www.mitre.org/work/tech_papers/tech_papers_99/peer_interaction/peer_interaction.pdf`. – Zugriffsdatum: 15.03.2006

[Soller & Lesgold 1999] Soller, Amy ; Lesgold, Alan: *Analyzing Peer Dialogue from an Active Learning Perspective*. 1999. – URL `http://cbl.leeds.ac.uk/~tamsin/dialogueworkshop/soller-peer-dialogue.pdf`. – Zugriffsdatum: 25.02.2003

[Soller et al. 2002] Soller, Amy ; Wiebe, Janyce ; Lesgold, Alan: *A Machine Learning Approach to Assessing Knowledge Sharing During Collaborative Learning Activities*. 2002. – URL `http://newmedia.colorado.edu/cscl/129.pdf`. – Zugriffsdatum: 30.03.2006

[Soller 2001] Soller, Amy: Supporting Social Interaction in an Intelligent Collaborative Learning System. In: *International Journal of Artificial Intelligence in Education* (2001), S. 40–62. – URL `http://computing.unn.ac.uk/staff/cgpb4/ijaied/members01/archive/vol\_12/soller/paper.pdf`. – Zugriffsdatum: 05.01.2004

[Soller 2004] Soller, Amy: *Computational Modeling and Analysis of Knowledge Sharing in Collaborative Distance Learning*. 2004. – URL `http://www.springerlink.com/app/home/contribution.asp?wasp=4e83c8ae0bd342aeab35ba1f4d75b13a\&referrer=parent\&backto=issue,3,3;journal,3,26;linkingpublicationresults,1:100343,1`. – Zugriffsdatum: 21.06.2005

[Sommerville 2007] Sommerville, Ian: *Software Engineering*. 8th edition. Harlow, England : Addison-Wesley, 2007

[Spencer & Pruss 1995] Spencer, John ; Pruss, Adrian: *Top Teams – Der Königsweg zu mehr Flexibilität, Effizienz und Erfolg im Betrieb*. München : Knaur, 1995

[Stahl 2002] Stahl, Gerry: Groupware Goes to School. In: Haake, Jörg M. (Hrsg.) ; Pino, Jos'e A. (Hrsg.): *CRIWG* Bd. 2440, Springer, 2002, S. 7–24. – URL `http://www.cis.drexel.edu/faculty/gerry/mit/ch07.pdf`. – Zugriffsdatum: 23.01.2006. – ISBN 3-540-44112-3

[Steffen 1998] Steffen, Jorge: *HTML 4 Power!* 2. Düsseldorf : Sybex, 1998

[Stevens & Collins 1977] Stevens, Albert L. ; Collins, Allan: The goal structure of a socratic tutor. In: *ACM '77: Proceedings of the 1977 annual conference*. New York, NY, USA : ACM Press, 1977, S. 256–263

[Stroustrup 1998] Stroustrup, Bjarne: *Die C++-Programmiersprache*. 3., aktualisierte und erweiterte Auflage. München : Addison-Wesley, 1998. – ISBN 3-8273-1296-5

[Tanenbaum 2003a] Tanenbaum, Andrew S.: *Computernetzwerke*. 4., überarbeitete Auflage. München : Prentice Hall, 2003. – ISBN 3-8273-7046-9

[Tanenbaum 2003b] Tanenbaum, Andrew S.: *Moderne Betriebssysteme*. 2., überarbeitete Auflage. München : Prentice Hall, 2003. – ISBN 3-8273-7019-1

[Teufel et al. 1995] Teufel, Stephanie ; Sauter, Christian ; Mühlherr, Thomas ; Bauknecht, Kurt: *Computerunterstützung für Gruppenarbeit*. Addison-Wesley, 1995

[Vizcaíno-Barceló 2002] Vizcaíno-Barceló, Aurora: *A Simulated Student Model for Improving Collaborative Learning.* 2002. – URL `http://archive-edutice.ccsd.cnrs.fr/docs/00/02/71/14/PDF/Vizca%EDno.pdf`. – Zugriffsdatum: 01.03.2006

[Vizcaíno & Du Boulay 2002] Vizcaíno, Aurora ; Du Boulay, Benedict: Using a Simulated Student to Repair Difficulties in Collaborative Learning. In: *Proceedings of ICCE'2002* (2002). – URL `http://www.cogs.susx.ac.uk/users/bend/papers/icce2002.pdf`. – Zugriffsdatum: 13.03.2006

[Vizcaino 2005] Vizcaino, Aurora: A Simulated Student Can Improve Collaborative Learning. In: *International Journal of Artificial Intelligence in Education* 15 (2005), S. 3–40. – URL `http://aied.inf.ed.ac.uk/abstract/Vol_15/Vizcaino05.html`. – Zugriffsdatum: 11.07.2006

[Vogel 1979] Vogel, Friedrich: *Beschreibende und schließende Statistik – Formeln, Definitionen, Erläuterungen, Stichwörter und Tabellen.* München : Oldenbourg, 1979. – ISBN 3-486-23451-X

[Vygotsky 1978] Vygotsky, Lev S.: *Mind in society.* Cambridge : Harvard University Press, 1978

[Wagner 2005] Wagner, Hartmut: *TMS – das Team Manangement System.* 2005. – URL `http://www.tms-zentrum.de/`. – Zugriffsdatum: 23.11.2005

[Wang et al. 2005] Wang, Ning ; Johnson, W. L. ; Rizzo, Paola ; Shaw, Erin ; Mayer, Richard E.: Experimental evaluation of polite interaction tactics for pedagogical agents. In: *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces.* New York, NY, USA : ACM Press, 2005, S. 12–19. – ISBN 1-58113-894-6

[Weidmann 2001] Weidmann, Heike: *E-Learning für den Mittelstand Outsourcing oder eigene Kompetenz aufbauen?* Kap. 3.1.1. Siehe (Hohenstein & Wilbers, 2005)

[Weizenbaum 1966] Weizenbaum, Joseph: ELIZA - A computer program for the study of natural language communication between man and machine. In: *Commun. ACM* 9 (1966), Nr. 1, S. 36–45. – URL `http://portal.acm.org/ft_gateway.cfm?id=365168&type=pdf&coll=ACM&dl=ACM&CFID=14429560&CFTOKEN=23725835`. – Zugriffsdatum: 15.02.2007. – ISSN 0001-0782

[Wessner & Pfister 2001] Wessner, Martin ; Pfister, Hans-Rüdiger: Group formation in computer-supported collaborative learning. In: *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, ACM Press, 2001, S. 24–31. – URL `http://doi.acm.org/10.1145/500286.500293`. – Zugriffsdatum: 14.02.2003. – ISBN 1-58113-294-8

[Wessner 2001] Wessner, Martin: *Software für e-Learning: Kooperative Umgebungen und Werkzeuge*. S. 195–219. Siehe (Schulmeister, 2001)

[Wessner 2004] Wessner, Martin: *Lerngruppen*. Kap. 3.1, S. 202–207. Siehe (Haake et al., 2004)

[Wessner 2005] Wessner, Martin: *Kontextuelle Kooperation in virtuellen Lernumgebungen*, TU Darmstadt, Dissertation, 2005. – URL `http://www.wessner.net/Dissertation-Wessner-2005.pdf`. – Zugriffsdatum: 28.11.2007

[Wilbers 2001] Wilbers, Karl: *E-Learning didaktisch gestalten*. Kap. 4.0. Siehe (Hohenstein & Wilbers, 2005)

[Winkler & Mandl 2003] Winkler, Katrin ; Mandl, Heinz: *Knowledge Master: Ein Blended-Learning Weiterbildungskonzept*. Kap. 5, S. 191–202. Siehe (Dittler, 2003b)

[Wittig 2002] Wittig, Frank: *Maschinelles Lernen Bayes'scher Netze für benutzeradaptive Systeme*, Universität des Saarlandes, Dissertation, 2002. – URL `http://deposit.ddb.de/cgi-bin/dokserv?idn=972323384\&dok\_var=d1\&dok\_ext=pdf\&filename=972323384.pdf`. – Zugriffsdatum: 07.02.2007

[Zimbardo & Gerrig 2000] Zimbardo, Philip G. ; Gerrig, Richard J.: *Psychologie*. 7. Auflage. Berlin : Springer, 2000

[Zuser et al. 2004] Zuser, Wolfgang ; Grechenig, Thomas ; Köhle, Monika: *Software Engineering mit UML und dem Unified Process*. 2., überarbeitete Auflage. München : Pearson Studium, 2004. – ISBN 3-8273-7090-6